Dodek Duo

Resolução de um Problema de Decisão usando Programação em Lógica com Restrições

Bernardo Leite e Francisco Silva, grupo: Dodek Duo_2

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-45 Porto, Portugal

Resumo

No âmbito da Unidade Curricular de Programação e Lógica e do trabalho prático número 2 “Resolução de um Problema de Decisão/Otimização usando Programação em Lógica com Restrições”, apresentamos este relatório para descrever e explicar todos os aspetos do nosso trabalho.

Os objetivos para este trabalho foram cumpridos, ou seja, tanto a resolução dos dois desafios inerentes ao Puzzle Dodek Duo bem como a geração dinâmica de Problemas, também em PLR, possibilitando a visualização dos Problemas e Soluções em diferentes contextos e complexidades.

Introdução

Através deste artigo pretendemos reportar todo o percurso que levamos para conseguir implementar o nosso Programa, detalhar toda a sua estrutura, caracterizar o seu funcionamento e ainda fundamentar as decisões que tomamos.

23 de Dezembro de 2017

A disposição do relatório segue a estrutura recomendada:

2. Descrição do Problema, para descrever com detalhe o Problema/Puzzle que o nosso grupo se propôs a resolver;

3. Abordagem, para descrever o método adotado na modelação do problema;

3.1 Variáveis de Decisão, em que descrevemos as variáveis de decisão utilizadas bem como os seus domínios;

3.2 Restrições, para descrever todas as restrições utilizadas ao longo do programa;

3.3 Estratégia de Pesquisa, para descrever a estratégia de etiquetagem (labeling) utilizada;

4. Visualização da solução, para explicar os predicados responsáveis por imprimir os enunciados e soluções dos problemas;

5. Resultados para demonstrar exemplos de aplicação em instâncias do problema com diferentes abordagens e analisar os resultados obtidos;

6. Conclusões e Trabalho Futuro, para refletir, concluir e sintetizar a informação das secções anteriores.

2. Descrição do Problema

O Dodek Duo tem a geometria de um Dodecaedro Regular e é composto por 12 faces pentagonais ligadas entre si. Cada face pentagonal é dividida em 5 triângulos com diferentes cores, portanto, são cinco cores que ocorrem em cada uma das faces. Note-se que a posição destas cores , em cada face, aparece numa ordem específica, de tal forma que todas as faces são únicas. O objetivo é resolver o Puzzle de maneira a que todas as faces adjacentes tenham a mesma cor.

Existe um outro desafio, que é quando as faces contêm figuras geométricas (semi-círculo, triângulo e retângulo) sendo que estas ocorrem uma ou duas vezes em cada face mas em diferentes ordens ou quantidades de maneira a que, mais uma vez, cada face seja única. O objetivo aqui é fazer com que todas as faces adjacentes tenham a mesma figura geométrica.

3. Abordagem

A nossa abordagem ao modelar os Problemas que dizem respeito a este trabalho podem ser representados por este esquema:

Dodek Duo			
Dodek Duo Puzzle 1 - Sol	Dodek Duo Puzzle 1 - Gen	Dodek Duo Puzzle 2 - Sol	Dodek Duo Puzzle 2 - Gen
Em dodekduo_Puzzle1_Sol.pl é o ficheiro onde estão implementas as restrições , variáveis de decisão e pesquisa para o primeiro problema original, o das cores.	Em dodekduo_Puzzle1_Gen.pl é o ficheiro onde se implementa a estratégia que gera problemas (enunciados) para o primeiro Puzzle em diferentes contextos. É garantida pelo menos uma solução para cada enunciado.	Em dodekduo_Puzzle2_Sol.pl é o ficheiro onde estão implementadas as restrições , variáveis de decisão e pesquisa para o segundo problema original, o das formas gemétricas.	Em dodekduo_Puzzle2_Gen.pl é o ficheiro onde se implementa a estratégia que gera problemas (enunciados) do segundo Puzzle em diferentes contextos. É garantida pelo menos uma solução para cada enunciado.

A nosso ver, consideramos que esta estruturação é pertinente para o que pretendemos realizar tendo em conta que existem diferenças significativas no que diz respeito à obtenção de uma solução ou à geração de um problemas/enunciados.

Nota: Este ponto (3) está dividido em duas partes, uma parte relativa à abordagem na obtenção de soluções e outra, para a geração dinâmica de problemas em diferentes contextos.

Parte 1: Obtenção de Soluções do Problema

3.1 Variáveis de Decisão

Relativamente às variáveis de decisão do nosso programa tomamos as seguintes decisões:

Os triângulos que compõem uma face são representados por cinco variáveis ***X1,X2,X3,X4,X5*** em que ***X*** é a face;

Uma face Pentagonal é representada por uma lista denominada ***Face i*** em que ***i*** é o número da face (i >=1 && i <=12);

O Dodecaedro é representado por uma **lista de listas**. Assim, **cada elemento é uma lista que representa uma só face**. Ao todo, 12 faces;

```
%Applying domain from 1 to 5 referi

domain(Face1,MinColor,MaxColor),
domain(Face2,MinColor,MaxColor),
domain(Face3,MinColor,MaxColor),
domain(Face4,MinColor,MaxColor),
domain(Face5,MinColor,MaxColor),
domain(Face6,MinColor,MaxColor),
domain(Face7,MinColor,MaxColor),
domain(Face8,MinColor,MaxColor),
domain(Face9,MinColor,MaxColor),
domain(Face10,MinColor,MaxColor),
domain(Face11,MinColor,MaxColor),
domain(Face12,MinColor,MaxColor),
```

```
Face1=[A1,A2,A3,A4,A5],
Face2=[B1,B2,B3,B4,B5],
Face3=[C1,C2,C3,C4,C5],
Face4=[D1,D2,D3,D4,D5],
Face5=[E1,E2,E3,E4,E5],
Face6=[F1,F2,F3,F4,F5],
Face7=[G1,G2,G3,G4,G5],
Face8=[H1,H2,H3,H4,H5],
Face9=[I1,I2,I3,I4,I5],
Face10=[J1,J2,J3,J4,J5],
Face11=[L1,L2,L3,L4,L5],
Face12=[M1,M2,M3,M4,M5],
```

```
AllFaces=[Face1,Face2,Face3,Face4,Face5,Face6,Face7,Face8,Face9,Face10,Face11,Face12],
```

Imagem 1 - Variáveis de Decisão utilizadas.

3.2 Restrições

Relativamente às restrições do nosso programa e tendo em vista a sua necessidade para que os requisitos do problema sejam cumpridos na integra, tomamos as seguintes decisões:

-> Fazer a restrição que faz com que **todas as cores de uma face sejam diferentes** - utilização do Predicado ***all_different/1*** (imagem 2 canto superior esquerdo);

-> Fazer a restrição que faz com que todas as **faces adjacentes tenham a mesma cor** - utilização do operador de restrição aritmético **#=** e preposicional **#/** (img 2 canto sup direito);

-> Fazer a restrição que **possibilita a cada uma das faces ser um dos 12 Pentágonos do enunciado inclusive todas as suas rotações**. Tendo em conta que são no total 12 Pentágonos e que cada pentágono tem 5 rotações possíveis, no total são 60 Pentágonos. Foi utilizado o predicado ***table/2*** (imagem 2 canto inferior esquerdo);

-> Fazer a restrição que obriga a que as **12 Faces que compõem o dodecaedro sejam diferentes** no que diz respeito à ordem em que as cores se apresentam (img 2 canto inf direito).

```
%Check if all the triangles from a face have different colors

all_different(Face1),
all_different(Face2),
all_different(Face3),
all_different(Face4),
all_different(Face5),
all_different(Face6),
all_different(Face7),
all_different(Face8),
all_different(Face9),
all_different(Face10),
all_different(Face11),
all_different(Face12),
```

```
A1#=B1 #/\ A2#=F1 #/\ A3#=E1 #/\ A4#=D1 #/\ A5#=C1 #/\

B1#=A1 #/\ B2#=C5 #/\ B3#=H5 #/\ B4#=G1 #/\ B5#=F2 #/\

C1#=A5 #/\ C2#=D5 #/\ C3#=I5 #/\ C4#=H1 #/\ C5#=B2 #/\

D1#=A4 #/\ D2#=E5 #/\ D3#=J5 #/\ D4#=I1 #/\ D5#=C2 #/\

E1#=A3 #/\ E2#=F5 #/\ E3#=L5 #/\ E4#=J1 #/\ E5#=D2 #/\

F1#=A2 #/\ F2#=B5 #/\ F3#=G5 #/\ F4#=L1 #/\ F5#=E2 #/\

G1#=B4 #/\ G2#=H4 #/\ G3#=M5 #/\ G4#=L2 #/\ G5#=F3 #/\

H1#=C4 #/\ H2#=I4 #/\ H3#=M1 #/\ H4#=G2 #/\ H5#=B3 #/\

I1#=D4 #/\ I2#=J4 #/\ I3#=M2 #/\ I4#=H2 #/\ I5#=C3 #/\

J1#=E4 #/\ J2#=L4 #/\ J3#=M3 #/\ J4#=I2 #/\ J5#=D3 #/\

L1#=F4 #/\ L2#=G4 #/\ L3#=M4 #/\ L4#=J2 #/\ L5#=E3 #/\

M1#=H3 #/\ M2#=I3 #/\ M3#=J3 #/\ M4#=L3 #/\ M5#=G3,
```

```
table([Face1],AllPenta),
table([Face2],AllPenta),
table([Face3],AllPenta),
table([Face4],AllPenta),
table([Face5],AllPenta),
table([Face6],AllPenta),
table([Face7],AllPenta),
table([Face8],AllPenta),
table([Face9],AllPenta),
table([Face10],AllPenta),
table([Face11],AllPenta),
table([Face12],AllPenta),
```

```
allListsDifferent2([],[],[]).
allListsDifferent2([H1|T1],[H2|T2],[HB|TB]):-
    H1 #= H2 #<=> HB,
    allListsDifferent2(T1,T2,TB).

allListsDifferent1(_,[]).
allListsDifferent1(HA,[HB|T]):-
    length(HA,N),
    length(LB,N),
    domain(LB,0,1),
    sum(LB,#<,N),
    allListsDifferent2(HA,HB,LB),
    allListsDifferent1(HA,T).

allListsDifferent([_]).
allListsDifferent([HA,HB|T]):-
    allListsDifferent1(HA,[HB|T]),
    allListsDifferent([HB|T]).
```

Imagem 2 - Restrições Utilizadas.

3.3 Estratégia de Pesquisa

No que diz respeito à estratégia de etiquetagem foi utilizado o predicado ***labeling/2*** em que o segundo argumento é uma lista de listas representando as 12 faces do Dodecaedro. Assim, em ***Solution*** irá ser gerada uma possível solução a cada pesquisa (imagem 3).

```
append([Face1,Face2,Face3,Face4,Face5,Face6,Face7,Face8,Face9,Face10,Face11,Face12],Solution),
labeling([], Solution),
```

Imagem 3 - Utilização do labeling.

Parte 2: Geração dinâmica de Problemas - 3.1 Variáveis de Decisão

A geração dinâmica de Problemas utilizando programação com restrições requereu uma abordagem diferente. Antes, ao utilizarmos o predicado ***table/2***, no primeiro argumento colocava-se a ***Face*** e no segundo argumento, uma **lista de inteiros** com as possíveis representações que essa face poderia adquirir (os 12 pentágonos do enunciado). Como nesta fase o objetivo é gerar esses tais 12 Pentágonos, (Problema) já **não se pode utilizar uma lista de inteiros pré-definida** mas sim uma **lista com Variáveis de domínio**. Para tal, utilizou-se o predicado ***fd_dom/2***, um predicado que permite instanciar uma variável com um número inteiro através de uma variável de domínio.

Imagem 4 - Variáveis de Decisão utilizadas na geração de Problemas para o Puzzle das Cores.

```
PentaA=[PentaA1,PentaA2,PentaA3,PentaA4,PentaA5],
PentaB=[PentaB1,PentaB2,PentaB3,PentaB4,PentaB5],
PentaC=[PentaC1,PentaC2,PentaC3,PentaC4,PentaC5],
PentaD=[PentaD1,PentaD2,PentaD3,PentaD4,PentaD5],
PentaE=[PentaE1,PentaE2,PentaE3,PentaE4,PentaE5],
PentaF=[PentaF1,PentaF2,PentaF3,PentaF4,PentaF5],
PentaG=[PentaG1,PentaG2,PentaG3,PentaG4,PentaG5],
PentaH=[PentaH1,PentaH2,PentaH3,PentaH4,PentaH5],
PentaI=[PentaI1,PentaI2,PentaI3,PentaI4,PentaI5],
PentaJ=[PentaJ1,PentaJ2,PentaJ3,PentaJ4,PentaJ5],
PentaL=[PentaL1,PentaL2,PentaL3,PentaL4,PentaL5],
PentaM=[PentaM1,PentaM2,PentaM3,PentaM4,PentaM5],

domain(PentaA,MinColor,MaxColor),
domain(PentaB,MinColor,MaxColor),
domain(PentaC,MinColor,MaxColor),
domain(PentaD,MinColor,MaxColor),
domain(PentaE,MinColor,MaxColor),
domain(PentaF,MinColor,MaxColor),
domain(PentaG,MinColor,MaxColor),
domain(PentaH,MinColor,MaxColor),
domain(PentaI,MinColor,MaxColor),
domain(PentaJ,MinColor,MaxColor),
domain(PentaL,MinColor,MaxColor),
domain(PentaM,MinColor,MaxColor),
```

```
fd_dom(A1,Face_A1),fd_dom(A2,Face_A2),fd_dom(A3,Face_A3),fd_dom(A4,Face_A4),fd_dom(A5,Face_A5),
fd_dom(B1,Face_B1),fd_dom(B2,Face_B2),fd_dom(B3,Face_B3),fd_dom(B4,Face_B4),fd_dom(B5,Face_B5),
fd_dom(C1,Face_C1),fd_dom(C2,Face_C2),fd_dom(C3,Face_C3),fd_dom(C4,Face_C4),fd_dom(C5,Face_C5),
fd_dom(D1,Face_D1),fd_dom(D2,Face_D2),fd_dom(D3,Face_D3),fd_dom(D4,Face_D4),fd_dom(D5,Face_D5),
fd_dom(E1,Face_E1),fd_dom(E2,Face_E2),fd_dom(E3,Face_E3),fd_dom(E4,Face_E4),fd_dom(E5,Face_E5),
fd_dom(F1,Face_F1),fd_dom(F2,Face_F2),fd_dom(F3,Face_F3),fd_dom(F4,Face_F4),fd_dom(F5,Face_F5),
fd_dom(G1,Face_G1),fd_dom(G2,Face_G2),fd_dom(G3,Face_G3),fd_dom(G4,Face_G4),fd_dom(G5,Face_G5),
fd_dom(H1,Face_H1),fd_dom(H2,Face_H2),fd_dom(H3,Face_H3),fd_dom(H4,Face_H4),fd_dom(H5,Face_H5),
fd_dom(I1,Face_I1),fd_dom(I2,Face_I2),fd_dom(I3,Face_I3),fd_dom(I4,Face_I4),fd_dom(I5,Face_I5),
fd_dom(J1,Face_J1),fd_dom(J2,Face_J2),fd_dom(J3,Face_J3),fd_dom(J4,Face_J4),fd_dom(J5,Face_J5),
fd_dom(L1,Face_L1),fd_dom(L2,Face_L2),fd_dom(L3,Face_L3),fd_dom(L4,Face_L4),fd_dom(L5,Face_L5),
fd_dom(M1,Face_M1),fd_dom(M2,Face_M2),fd_dom(M3,Face_M3),fd_dom(M4,Face_M4),fd_dom(M5,Face_M5),
```

3.2 Restrições

Relativamente às Restrições para a geração de Problemas, utilizamos as mesmas que foram aplicadas para quando o objetivo era encontrar a solução. A única diferença agora é, mais uma vez, ao utilizar o predicado ***table/2*** **não se sabe à priori quais os Pentágonos** (***AllFaces_VAR***) que podem ser utilizados pois isso é precisamente o que se pretende descobrir (**gerar**).

```
table([PentaA],AllFaces_VAR),
table([PentaB],AllFaces_VAR),
table([PentaC],AllFaces_VAR),
table([PentaD],AllFaces_VAR),
table([PentaE],AllFaces_VAR),
table([PentaF],AllFaces_VAR),
table([PentaG],AllFaces_VAR),
table([PentaH],AllFaces_VAR),
table([PentaI],AllFaces_VAR),
table([PentaJ],AllFaces_VAR),
table([PentaL],AllFaces_VAR),
table([PentaM],AllFaces_VAR),
```

```
all_different(Face1),
all_different(Face2),
all_different(Face3),
all_different(Face4),
all_different(Face5),
all_different(Face6),
all_different(Face7),
all_different(Face8),
all_different(Face9),
all_different(Face10),
all_different(Face11),
all_different(Face12),
```

Imagem 5 - Restrições utilizadas na implementação das geração de Problemas.

3.3 Estratégia de Pesquisa

No que diz respeito à estratégia de etiquetagem foi utilizado o predicado ***labeling/2*** em que o segundo argumento é uma **lista de listas** que representam as 12 faces do Dodecaedro. Desta forma, em ***Solution*** irá ser gerada uma **possível solução a cada pesquisa** e para além disso, também irá ser gerada a **lista de pentágonos usada** para esse efeito.

```
append([PentaA,PentaB,PentaC,PentaD,PentaE,PentaF,PentaG,PentaH,PentaI,PentaJ,PentaL,PentaM,
Face1,Face2,Face3,Face4,Face5,Face6,Face7,Face8,Face9,Face10,Face11,Face12],Solution),
labeling([], Solution),
```

Imagem 6 - Utilização do labeling para gerar Faces (Solução) e respetivos Pentágonos (Enunciado).

Puzzle 2- No que diz respeito à obtenção de Soluções

Todos os passos descritos até agora são referentes ao puzzle 1, relativo à junção das faces através das cores. Os mesmos passos, em termos de implementação, foram dados para o puzzle 2 em que o objetivo é juntar as faces com as figuras geométricas. Porém, existe uma diferença, é que neste caso já não se utiliza o predicado ***all_different/1*** visto que em cada face pode haver repetição das figuras, no máximo duas repetições.

Puzzle 2- No que diz respeito à geração de Problemas

Existem diferenças significativas no que toca à geração de Problemas/Enunciados para o Puzzle 2. Neste caso, cada uma das faces terá de conter três figuras geométricas possibilitando repetições das mesmas e, por isso, foi implementado no predicado ***forceShapes/3*** que obriga a que em cada face exista uma das três figuras geométricas. Para além disso, estipula-se que em cada face **existe a obrigatoriedade de cada figura aparecer 1 ou 2 vezes,** e para isso utiliza-se o predicado ***count/4***.

```
checkAppears1PerFace(AllElements,Figure):-

    nth1(1,AllElements,Face1), count(Figure,Face1,#=,Count1), Count1#>=1,Count1#=<2,
    nth1(2,AllElements,Face2), count(Figure,Face2,#=,Count2), Count2#>=1,Count2#=<2,
    nth1(3,AllElements,Face3), count(Figure,Face3,#=,Count3), Count3#>=1,Count3#=<2,
    nth1(4,AllElements,Face4), count(Figure,Face4,#=,Count4), Count4#>=1,Count4#=<2,
    nth1(5,AllElements,Face5), count(Figure,Face5,#=,Count5), Count5#>=1,Count5#=<2,
    nth1(6,AllElements,Face6), count(Figure,Face6,#=,Count6), Count6#>=1,Count6#=<2,
    nth1(7,AllElements,Face7), count(Figure,Face7,#=,Count7), Count7#>=1,Count7#=<2,
    nth1(8,AllElements,Face8), count(Figure,Face8,#=,Count8), Count8#>=1,Count8#=<2,
    nth1(9,AllElements,Face9), count(Figure,Face9,#=,Count9), Count9#>=1,Count9#=<2,
    nth1(10,AllElements,Face10), count(Figure,Face10,#=,Count10), Count10#>=1,Count10#=<2,
    nth1(11,AllElements,Face11), count(Figure,Face11,#=,Count11), Count11#>=1,Count11#=<2,
    nth1(12,AllElements,Face12), count(Figure,Face12,#=,Count12), Count12#>=1,Count12#=<2.

forceShapes(AllElements,MaxFigures,Counter):-Counter=MaxFigures.
forceShapes(AllElements,MaxFigures,Counter):-
    Counter<MaxFigures,
    checkAppears1PerFace(AllElements,Counter),
    NewCounter is Counter+1,
forceShapes(AllElements,MaxFigures,NewCounter).
```

Imagem 7 - Utilização do Predicado count/4.

4. Visualização

Existem dois Predicados responsáveis por visualizar a solução em modo de texto. São eles o ***printProblem/1*** e ***printSolution/2***. O primeiro mostra os Pentágonos do Problema a resolver, ou seja, os Pentágonos com os quais o utilizador poderá obter uma solução, já o segundo Predicado imprime a solução, e para além disso, **para cada face mostra a referência ao Pentágono utilizado** como se poderá ver mais à frente na secção de Resultados.

```
iterateSolution(Solution,Problem,Size,Counte
iterateSolution(Solution,Problem,Size,Counte
    Counter<Size,
    Counter2 is Counter+1, Counter3 is Coun
    Counter4 is Counter+3, Counter5 is Coun
    nth1(Counter,Solution,Elem1),nth1(Counte
    nth1(Counter3,Solution,Elem3),nth1(Coun
    nth1(Counter5,Solution,Elem5),
    Face=[Elem1,Elem2,Elem3,Elem4,Elem5],
    discoverPenta(Face,Problem,1,Penta),
    write('Face '), write(Numerator), write
    NewCounter is Counter + 5,
    NewNumerator is Numerator+1,
iterateSolution(Solution,Problem,Size,NewCo

printSolution(Problem,Solution):-
    length(Solution,S),
    Size is S+1,
    iterateSolution(Solution,Problem,Size,1,1).
```

```
iterateProblem(Problem,[],Size,Counter):- Counter=Size.
iterateProblem(Problem,[],Size,Counter):-
    Counter<Size,
    nth1(Counter,Problem,Penta),
    write('Penta '), write(Counter), write(' = '), write(Per
    NewCounter is Counter + 1,
iterateProblem(Problem,[],Size,NewCounter).

printProblem(Problem):-
    length(Problem,S),
    Size is S+1,
    iterateProblem(Problem,[],Size,1).
```

Imagem 8 - Implementação dos predicados responsáveis por imprimir os Problemas e Soluções em modo de texto.

A chamada destes Predicados é feita quando se obtém uma solução para um dado Problema, depois de se utilizar o predicado **labeling/2**:

```
write('**********Problem Input: 12 Pentagons**********'), nl,
write('-> Nr of Colors: '), write(MaxColor),nl,
generateProb(AllFaces,Problem),printProblem(Problem),nl,nl,

write('-> Note: Because there are 60 rotational symmetries '),nl,
write('in a regular dodecahedron there are (x solutions * 60) total solutions to this problem.'),nl,nl,

write('-> Want to see a possible Solution? (Type 1 for YES and 2 for NO.)'),nl,
write('Your Option (Select 1 or 2): '), nl, read(Option),nl,nl,

if_then_else(Option=1,
(write('**********Solution:**********'), nl,
printSolution(Problem,Solution),nl,nl), menu),

write('-> Want to see More Solutions? (Type 1 for YES and 2 for NO.)'),nl,
write('Your Option (Select 1 or 2): '), nl, read(Option3),nl,nl,

if_then_else(Option3=1,dodekduo_Puzzle1_Sol(Problem,NewSolution,MinColor,MaxColor,1), continueplay),

write('-> Generate other Problem? (Type 1 for YES and 2 for NO)'),nl,
write('Your Option (Select 1 or 2): '), nl, read(Option2),nl,nl,
```

Imagem 9 - Chamada dos predicados responsáveis por imprimir as Soluções em modo de texto.

5. Resultados

Antes de demonstrarmos alguns exemplos de Resultados, consideramos pertinente sublinhar uma característica inerente ao sólido geométrico deste Puzzle, um Dodecaedro Regular. Relativamente ao primeiro problema, sabíamos à priori que existia apenas uma solução e no segundo, 47 soluções. Aquando das observação dos resultados, o número de soluções que estávamos a obter eram 60 e 2820, respetivamente. Ora, o que na altura nos intrigou ficou rapidamente esclarecido porque logo nos apercebemos que um **Dodecaedro Regular tem 60 simetrias de rotação.** Desta forma, para o primeiro problema existe 1 solução * 60 simetrias = 60 soluções efectivas e para o segundo, 47 soluções * 60 simetrias = 2820 soluções efectivas.

Um outro ponto importante de referir é que na apresentação dos resultados, as soluções são apresentadas com as Faces numeradas de 1 a 12. Esta numeração é feita na perspetiva **Topo-Cima e da Esquerda-Direita em relação à seguinte figura:**

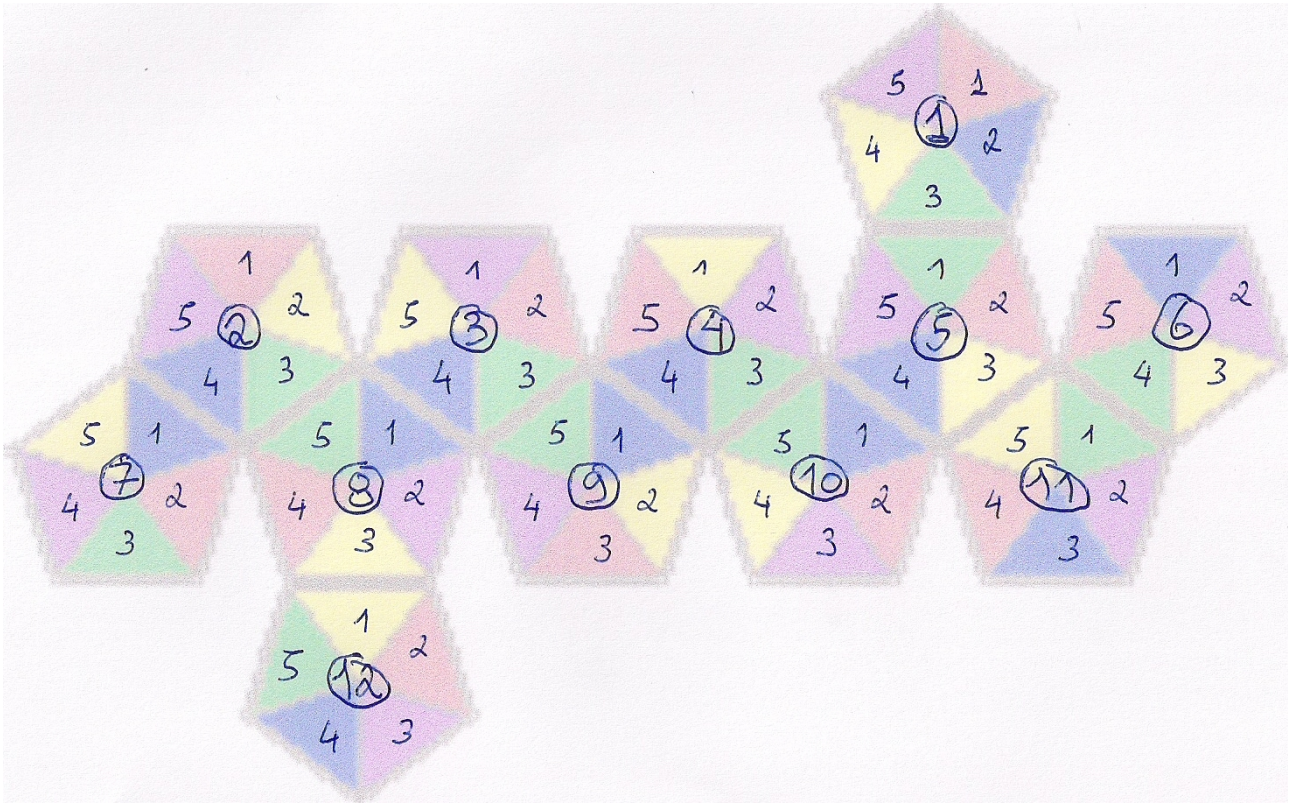

Imagem 10 - Este imagem ilustra o tipo de numeração feito no Dodecaedro. Portanto, nas soluções seguintes o número das Faces representam os números que estão centrados em cada Pentágono desta figura. Para além disso, a ordem pela qual os triângulos estão numerados é a mesma ordem que aparece na apresentação das soluções.

Os valores dentro das Faces e Pentágonos representam cores (Puzzle 1) ou Figuras (Puzzle 2):

1 - Laranja | 2 - Azul | 3 - Verde | 4 - Amarelo | 5 - Rosa……. 1 - Retângulo | 2 - Semi-Círculo | 2 - Retângulo

Obtenção de Soluções para o Primeiro Problema (1 solução)

Primeiro Passo: Seleção do Puzzle 1

```
***Welcome to Dodek Duo Puzzle!***

There are two Puzzles:
Puzzle 1: The aim is to assemble the puzzle so that at every edge two triangles of the same colour meet.
Puzzle 2: The aim is to assemble the puzzle so that at every edge identical shapes meet.
What do you want to do?

Options:
(1) -> Solve Puzzle 1 (Same Color Meet)
(2) -> Solve Puzzle 2 (Same Shapes Meet)
Your Option (Select 1 or 2):
|: 1.
```

Segundo Passo: Selecionar o Problema Original 1

```
How do you want to do with this Puzzle?:
(1) -> See Solution(s) of the Original Game.

(2) -> Randomly generate a problem to be solved.
```

Terceiro Passo: Obtenção da Solução

```
**********Problem Input: 12 Pentagons**********

Penta 1 = [1,2,5,4,3]
Penta 2 = [1,5,3,2,4]
Penta 3 = [4,5,1,3,2]
Penta 4 = [4,3,5,2,1]
Penta 5 = [2,3,4,1,5]
Penta 6 = [1,2,3,4,5]
Penta 7 = [3,5,4,2,1]
Penta 8 = [4,1,3,2,5]
Penta 9 = [3,2,1,5,4]
Penta 10 = [1,4,2,5,3]
Penta 11 = [2,5,1,4,3]
Penta 12 = [4,5,3,2,1]


-> Note: Because there are 60 rotational symmetries
in a regular dodecahedron there are (1 solution * 60) total solutions to this problem.

**********Solution:**********

Face 1 = [1,2,3,4,5] ... Penta Used : (6)
Face 2 = [1,4,3,2,5] ... Penta Used : (11)
Face 3 = [5,1,3,2,4] ... Penta Used : (3)
Face 4 = [4,5,3,2,1] ... Penta Used : (12)
Face 5 = [3,1,4,2,5] ... Penta Used : (10)
Face 6 = [2,5,4,3,1] ... Penta Used : (1)
Face 7 = [2,1,3,5,4] ... Penta Used : (7)
Face 8 = [2,5,4,1,3] ... Penta Used : (8)
Face 9 = [2,4,1,5,3] ... Penta Used : (2)
Face 10 = [2,1,5,4,3] ... Penta Used : (9)
Face 11 = [3,5,2,1,4] ... Penta Used : (4)
Face 12 = [4,1,5,2,3] ... Penta Used : (5)

-> Want to see other possible Solution? (Type 1 for YES and 2 for NO.)
Your Option (Select 1 or 2):
|:
```

Imagem 11 - Solução obtida para o Primeiro Problema, os números (1-5) representam as cores. Note-se que à direita de cada Face encontra-se o Pentágono utilizado. Este Pentágono poderá ter sofrido uma rotação.

Obtenção de Soluções para o Segundo Problema (47 soluções)

Primeiro Passo: Seleção do Puzzle 2

```
***Welcome to Dodek Duo Puzzle!***

There are two Puzzles:
Puzzle 1: The aim is to assemble the puzzle so that at every edge two triangles of the same colour meet.
Puzzle 2: The aim is to assemble the puzzle so that at every edge identical shapes meet.
What do you want to do?

Options:
(1) -> Solve Puzzle 1 (Same Color Meet)
(2) -> Solve Puzzle 2 (Same Shapes Meet)
Your Option (Select 1 or 2):
|: 1.
```

Segundo Passo: Selecionar o Problema Original 1

```
How do you want to do with this Puzzle?:
(1) -> See Solution(s) of the Original Game.

(2) -> Randomly generate a problem to be solved.
```

Terceiro Passo: Obtenção da Solução

```
Searching for all 47 solutions...


**********Problem Input: 12 Pentagons**********

Penta 1 = [1,2,3,3,1]
Penta 2 = [1,2,1,3,3]
Penta 3 = [3,1,2,2,3]
Penta 4 = [3,2,2,1,1]
Penta 5 = [3,3,2,2,1]
Penta 6 = [3,1,1,2,2]
Penta 7 = [2,3,2,1,1]
Penta 8 = [2,3,3,2,1]
Penta 9 = [3,3,2,1,1]
Penta 10 = [3,1,1,3,2]
Penta 11 = [2,3,2,1,3]
Penta 12 = [1,2,1,2,3]


-> Note: Because there are 60 rotational symmetries
in a regular dodecahedron there are (x solution * 60) total solutions to this problem.

-> The 47 solutions presented here correspond to the problem statement.

***Solution nr: 1***

Face 1 = [1,2,3,3,1] ... Penta Used : (1)
Face 2 = [1,1,2,2,3] ... Penta Used : (6)
Face 3 = [1,3,2,3,1] ... Penta Used : (10)
Face 4 = [3,1,2,1,3] ... Penta Used : (2)
Face 5 = [3,3,2,1,1] ... Penta Used : (9)
Face 6 = [2,3,2,1,3] ... Penta Used : (11)
Face 7 = [2,1,1,3,2] ... Penta Used : (4)
Face 8 = [3,3,2,1,2] ... Penta Used : (8)
Face 9 = [1,1,2,3,2] ... Penta Used : (7)
Face 10 = [1,2,3,1,2] ... Penta Used : (12)
Face 11 = [1,3,3,2,2] ... Penta Used : (5)
Face 12 = [2,2,3,3,1] ... Penta Used : (3)


-> Want to see next Solution? (Type 1 for YES and 2 for NO.)
Your Option (Select 1 or 2):
|:
```

Imagem 12 - Solução obtida para o Segundo Problema, os números (1-3) representam as figuras geométricas. Note-se que à direita de cada Face encontra-se o Pentágono utilizado. Este Pentágono poderá ter sofrido uma rotação.

Obtenção de Problemas/Enunciados

Obtenção de Problemas para o Puzzle das Cores

É possível gerar problemas com várias complexidades ao variar o número de cores, ou seja, no total podem existir no Dodecaedro mais de 5 cores. A nossa abordagem permite gerar vários problemas dando um certo número de cores. Só é permitido gerar 5 a 12 cores pelos seguintes motivos:

-> Com menos de cinco cores o Puzzle perderia o seu conceito, ou seja, a obrigatoriedade de em cada uma das faces existir cores diferentes;

-> Não mais de 12 cores pelo facto do Dodecaedro ter apenas 12 faces;

```
**********Problem Input: 12 Pentagons**********
-> Nr of Colors: 10
Penta 1 = [4,2,5,1,3]
Penta 2 = [1,7,2,3,5]
Penta 3 = [1,3,5,8,9]
Penta 4 = [6,3,5,1,2]
Penta 5 = [1,2,3,4,5]
Penta 6 = [2,3,5,1,4]
Penta 7 = [10,2,6,7,9]
Penta 8 = [6,10,8,4,5]
Penta 9 = [3,4,6,1,2]
Penta 10 = [6,3,4,1,2]
Penta 11 = [6,1,3,4,2]
Penta 12 = [3,1,4,5,2]


-> Note: Because there are 60 rotational symmetries
in a regular dodecahedron there are (x solutions * 60) total solutions to this problem.

-> Want to see a possible Solution? (Type 1 for YES and 2 for NO.)
Your Option (Select 1 or 2):
|: 1.


**********Solution:**********
Face 1 = [1,2,3,4,5] ... Penta Used : (5)
Face 2 = [1,2,3,4,6] ... Penta Used : (9)
Face 3 = [5,1,3,4,2] ... Penta Used : (1)
Face 4 = [4,2,3,5,1] ... Penta Used : (6)
Face 5 = [3,1,4,5,2] ... Penta Used : (12)
Face 6 = [2,6,3,5,1] ... Penta Used : (4)
Face 7 = [4,1,2,6,3] ... Penta Used : (10)
Face 8 = [4,2,6,1,3] ... Penta Used : (11)
Face 9 = [5,1,7,2,3] ... Penta Used : (2)
Face 10 = [5,8,9,1,3] ... Penta Used : (3)
Face 11 = [5,6,10,8,4] ... Penta Used : (8)
Face 12 = [6,7,9,10,2] ... Penta Used : (7)


-> Generate other Problem? (Type 1 for YES and 2 for NO)
Your Option (Select 1 or 2):
|:
```

Imagem 12 - Solução obtida para o Primeiro Problema (agora com 10 cores) , os números (1-10) representam as cores. Note-se que à direita de cada Face encontra-se o Pentágono utilizado. Este Pentágono poderá ter sofrido uma rotação.

Obtenção de Problemas para o Puzzle das Figuras Geométricas

É possível gerar problemas de várias complexidades ao variar o número das figuras, ou seja, no total podem existir no Dodecaedro mais de 3 figuras. A nossa abordagem permite gerar vários problemas dando um certo número de figuras. Só é permitido gerar 3 a 5 figuras pelos seguintes motivos:

-> Com menos de 3 figuras o Puzzle perderia o seu conceito, ou seja, a obrigatoriedade de em cada uma das faces existir pelo menos uma das figuras geométricas e ainda a possibilidade de essa figura se repetir na mesma face;

-> Não mais de 5 figuras porque, a partir desse valor, tornava-se um Problema igual ao das Cores;

```
**********Problem Input: 12 Pentagons**********
-> Nr of Colors: 4
Penta 1 = [3,1,2,4,2]
Penta 2 = [4,1,3,1,2]
Penta 3 = [4,1,2,3,1]
Penta 4 = [1,2,4,3,1]
Penta 5 = [3,2,4,1,1]
Penta 6 = [3,2,4,1,2]
Penta 7 = [1,3,3,2,4]
Penta 8 = [2,1,3,4,1]
Penta 9 = [4,1,2,3,3]
Penta 10 = [1,2,3,4,1]
Penta 11 = [4,3,1,2,3]
Penta 12 = [3,1,2,4,3]


-> Note: Because there are 60 rotational symmetries
in a regular dodecahedron there are (x solutions * 60) total solutions to this problem.

-> Want to see a possible Solution? (Type 1 for YES and 2 for NO.)
Your Option (Select 1 or 2):
|: 1.


**********Solution:**********
Face 1 = [1,1,2,3,4] ... Penta Used : (10)
Face 2 = [1,1,2,4,3] ... Penta Used : (4)
Face 3 = [4,1,2,3,1] ... Penta Used : (3)
Face 4 = [3,1,2,4,1] ... Penta Used : (2)
Face 5 = [2,1,3,4,1] ... Penta Used : (8)
Face 6 = [1,3,2,4,1] ... Penta Used : (5)
Face 7 = [4,1,2,3,2] ... Penta Used : (6)
Face 8 = [3,3,4,1,2] ... Penta Used : (9)
Face 9 = [4,1,3,3,2] ... Penta Used : (7)
Face 10 = [4,2,3,1,2] ... Penta Used : (1)
Face 11 = [4,3,1,2,3] ... Penta Used : (11)
Face 12 = [4,3,3,1,2] ... Penta Used : (12)


-> Generate other Problem? (Type 1 for YES and 2 for NO)
Your Option (Select 1 or 2):
|:
```

Imagem 13 - Solução obtida para o Segundo Problema (agora com 4 figuras), os números (1-4) representam as figuras geométricas. Note-se que à direita de cada Face encontra-se o Pentágono utilizado. Este Pentágono poderá ter sofrido uma rotação.

6. Conclusões e Trabalho Futuro

O teor deste projeto foi-nos complemente novo e, por isso, requereu da nossa parte uma estudo aprofundado sobre o assunto. Desta forma, e ultrapassando a fase inicial deste projeto onde ainda nos estávamos a contextualizar com a Programação com Restrições, progredimos positivamente para as fases seguintes. Apercebemo-nos da enorme vantagem que pode haver em Programar com Restrições porque pela primeira vez, aprendemos um método robusto e eficaz de resolver um problema do tipo Puzzle/Quebra-Cabeça. O facto de termos conseguido implementar um programa capaz de obter soluções para problemas sofisticados de forma tão rápida faz-nos afirmar que este foi dos trabalhos que nos deu mais gratificação de fazer. Para além disso, conseguimos também gerar novos Problemas com diferentes complexidades e para estes, soluções novas . Em suma, contamos com a experiência adquirida neste trabalho para aplicar esse conhecimento em desafios futuros ou até sermos nós mesmos a “inventar” um novo problema.

Bibliografia

https://sicstus.sics.se/documentation.html, SICstus

Programação em Lógica com Restrições, Henrique Lopes Cardoso

Programação em Lógica com Restrições no SICStus Prolog, Henrique Lopes Cardoso

Anexo

Código Fonte

```
:-use_module(library(clpfd)), use_module(library(lists)), use_module(library(random)).
:-include(dodekduo_Puzzle1_Sol).
:-include(dodekduo_Puzzle2_Sol).
:-include(dodekduo_Puzzle1_Gen).
:-include(dodekduo_Puzzle2_Gen).


playProblem1(Solution,Pentagons,5,0):-
    if_then_else(dodekduo_Puzzle1_Sol(Solution,Pentagons,5,0),continueplay,noMoreSol).


playProblem2(Solution,Pentagons,3,0):-
    if_then_else(dodekduo_Puzzle2_Sol(Solution,Pentagons,3,0),continueplay,noMoreSol).


forceColors(AllElements,MaxColor,Counter):-Counter=MaxColor.
forceColors(AllElements,MaxColor,Counter):-
    Counter<MaxColor,
    element(X,AllElements,Counter),
    NewCounter is Counter+1,
forceColors(AllElements,MaxColor,NewCounter).

noMoreSol:- nl,nl,write('!!!No More Solutions!!!'),nl,nl,
            menu.

checkAppears1PerFace(AllElements,Figure):-

    nth1(1,AllElements,Face1), count(Figure,Face1,#=,Count1), Count1#>=1,Count1#=<2,
    nth1(2,AllElements,Face2), count(Figure,Face2,#=,Count2), Count2#>=1,Count2#=<2,
    nth1(3,AllElements,Face3), count(Figure,Face3,#=,Count3), Count3#>=1,Count3#=<2,
    nth1(4,AllElements,Face4), count(Figure,Face4,#=,Count4), Count4#>=1,Count4#=<2,
    nth1(5,AllElements,Face5), count(Figure,Face5,#=,Count5), Count5#>=1,Count5#=<2,
    nth1(6,AllElements,Face6), count(Figure,Face6,#=,Count6), Count6#>=1,Count6#=<2,
    nth1(7,AllElements,Face7), count(Figure,Face7,#=,Count7), Count7#>=1,Count7#=<2,
    nth1(8,AllElements,Face8), count(Figure,Face8,#=,Count8), Count8#>=1,Count8#=<2,
    nth1(9,AllElements,Face9), count(Figure,Face9,#=,Count9), Count9#>=1,Count9#=<2,
    nth1(10,AllElements,Face10), count(Figure,Face10,#=,Count10),
    Count10#>=1,Count10#=<2,
    nth1(11,AllElements,Face11), count(Figure,Face11,#=,Count11),
    Count11#>=1,Count11#=<2,
    nth1(12,AllElements,Face12), count(Figure,Face12,#=,Count12),
    Count12#>=1,Count12#=<2.

forceShapes(AllElements,MaxFigures,Counter):-Counter=MaxFigures.
forceShapes(AllElements,MaxFigures,Counter):-
    Counter<MaxFigures,
    checkAppears1PerFace(AllElements,Counter),
    NewCounter is Counter+1,
forceShapes(AllElements,MaxFigures,NewCounter).

same([], []).

same([H1|R1], [H2|R2]):-
    H1 = H2,
    same(R1, R2).

iteratePenta(Solution,Aux,Size,Counter,Problem):- Counter=Size,
random_permutation(Aux,Problem).
iteratePenta(Solution,Aux,Size,Counter,Problem):-
    Counter<Size,
    nth0(Counter,Solution,Penta),
    random(0,4,Rot),
    shift(Penta,Rot,NewPenta),
    append(Aux,[NewPenta],NewAux),
    NewCounter is Counter+1,
iteratePenta(Solution,NewAux,Size,NewCounter,Problem).

iterateProblem(Problem,[],Size,Counter):- Counter=Size.
iterateProblem(Problem,[],Size,Counter):-
    Counter<Size,
    nth1(Counter,Problem,Penta),
    write('Penta '), write(Counter), write(' = '), write(Penta), nl,
```

```
69      NewCounter is Counter + 1,
70  iterateProblem(Problem,[],Size,NewCounter).
71
72  printProblem(Problem):-
73      length(Problem,S),
74      Size is S+1,
75      iterateProblem(Problem,[],Size,1).
76
77
78
79  discoverPenta(Face,Problem,Counter,Penta):-
80      nth1(Counter,Problem,CurrPent),
81
        shift(CurrPent,0,Res0),shift(CurrPent,1,Res1),shift(CurrPent,2,Res2),shift(CurrPen
        t,3,Res3),shift(CurrPent,4,Res4),
82
        if_then_else((same(Face,Res0);same(Face,Res1);same(Face,Res2);same(Face,Res3);same
        (Face,Res4)),
83                   Penta is Counter,
84                  (NewCounter is Counter+1,
                    discoverPenta(Face,Problem,NewCounter,Penta))).
85
86
87
88
89  iterateSolution(Solution,Problem,Size,Counter,Numerator):- Counter=Size.
90  iterateSolution(Solution,Problem,Size,Counter,Numerator):-
91      Counter<Size,
92      Counter2 is Counter+1, Counter3 is Counter+2, Counter4 is Counter+3, Counter5 is
        Counter + 4,
93
        nth1(Counter,Solution,Elem1),nth1(Counter2,Solution,Elem2),nth1(Counter3,Solution,
        Elem3),nth1(Counter4,Solution,Elem4),nth1(Counter5,Solution,Elem5),
94      Face=[Elem1,Elem2,Elem3,Elem4,Elem5],
95      discoverPenta(Face,Problem,1,Penta),
96      write('Face '), write(Numerator), write(' = '), write(Face), write(' ... Penta
        Used : ('), write(Penta), write(') '), nl,
97      NewCounter is Counter + 5,
98      NewNumerator is Numerator+1,
99  iterateSolution(Solution,Problem,Size,NewCounter,NewNumerator).
100
101 printSolution(Problem,Solution):-
102     length(Solution,S),
103     Size is S+1,
104     iterateSolution(Solution,Problem,Size,1,1).
105
106
107
108
109 generateProb(Solution,Problem):-
110     length(Solution,Size),
111     iteratePenta(Solution,[],Size,0,Problem).
112
113
114 allListsDifferent2([],[],[]).
115 allListsDifferent2([H1|T1],[H2|T2],[HB|TB]):-
116     H1 #= H2 #<=> HB,
117     allListsDifferent2(T1,T2,TB).
118
119 allListsDifferent1(_,[]).
120 allListsDifferent1(HA,[HB|T]):-
121     length(HA,N),
122     length(LB,N),
123     domain(LB,0,1),
124     sum(LB,#<,N),
125     allListsDifferent2(HA,HB,LB),
126     allListsDifferent1(HA,T).
127
128 allListsDifferent([_]).
129 allListsDifferent([HA,HB|T]):-
130     allListsDifferent1(HA,[HB|T]),
131     allListsDifferent([HB|T]).
```

```
132
133   shift(L1, N, L2) :-
134       N < 0, !,
135       N1 is -N,
136       shift(L2, N1, L1).
137
138   shift(L1, N, L2) :-
139       append(Lx, Ly, L1), % L1 is Lx || Ly
140       append(Ly, Lx, L2), % L2 is Ly || Lx
141       length(Lx, N).      % The length of Lx is N
142
143   generateAllPentagons(Pentagons,Tam,AuxList,Result,Counter):- Counter=Tam,
      Result=AuxList.
144
145   generateAllPentagons(Pentagons,Tam,AuxList,Result,Counter):-
146       Counter<Tam,
147       nth0(Counter,Pentagons,Penta),
148
      shift(Penta,0,Res),shift(Penta,1,Res1),shift(Penta,2,Res2),shift(Penta,3,Res3),shi
      ft(Penta,4,Res4),
149       append([[Res,Res1,Res2,Res3,Res4],AuxList],NewAuxList),
150       NewCounter is Counter+1,
151   generateAllPentagons(Pentagons,Tam,NewAuxList,Result,NewCounter).
152
153   getProblem1Penta(Pentagons):-
154       Penta1=[1,2,5,4,3],Penta2=[1,5,3,2,4],Penta3=[4,5,1,3,2],
155       Penta4=[4,3,5,2,1],Penta5=[2,3,4,1,5],Penta6=[1,2,3,4,5],
156       Penta7=[3,5,4,2,1],Penta8=[4,1,3,2,5],Penta9=[3,2,1,5,4],
157       Penta10=[1,4,2,5,3],Penta11=[2,5,1,4,3],Penta12=[4,5,3,2,1],
158
      Pentagons=[Penta1,Penta2,Penta3,Penta4,Penta5,Penta6,Penta7,Penta8,Penta9,Penta10,
      Penta11,Penta12].
159
160   getProblem2Penta(Pentagons):-
161       Penta1=[1,2,3,3,1],Penta2=[1,2,1,3,3],Penta3=[3,1,2,2,3],Penta4=[3,2,2,1,1],
162       Penta5=[3,3,2,2,1],Penta6=[3,1,1,2,2],Penta7=[2,3,2,1,1],Penta8=[2,3,3,2,1],
163       Penta9=[3,3,2,1,1],Penta10=[3,1,1,3,2],Penta11=[2,3,2,1,3],Penta12=[1,2,1,2,3],
164
      Pentagons=[Penta1,Penta2,Penta3,Penta4,Penta5,Penta6,Penta7,Penta8,Penta9,Penta10,
      Penta11,Penta12].
165
166
167   if_then_else(Condition, Action1, Action2) :- Condition, !, Action1.
168   if_then_else(Condition, Action1, Action2) :- Action2.
169
170   continueplay:- write('').
171   error:- write('!!!Please, check your input!!!'),menu.
172
173   menu:-  getProblem1Penta(Pentagons1),getProblem2Penta(Pentagons2),
174           nl,nl,nl,nl,nl,nl,nl,nl,nl,nl,nl,
175           write('***Welcome to Dodek Duo Puzzle!***') ,nl,nl,
176           write('There are two Puzzles:') ,nl,
177           write('Puzzle 1: The aim is to assemble the puzzle so that at every edge two
          triangles of the same colour meet.') ,nl,
178           write('Puzzle 2: The aim is to assemble the puzzle so that at every edge
          identical shapes meet.') ,nl,
179           write('What do you want to do?'), nl,nl,
180           write('Options:'),nl,
181           write('(1) -> Solve Puzzle 1 (Same Color Meet)'),nl,
182           write('(2) -> Solve Puzzle 2 (Same Shapes Meet)'),nl,
183           write('Your Option (Select 1 or 2): '), nl,read(Puzzle),nl,nl,
184           if_then_else((Puzzle=1;Puzzle=2), write('How do you want to do with this
          Puzzle?:'), continueplay) ,nl,
185           if_then_else(Puzzle=1, write('(1) -> See Solution(s) of the Original
          Game.'), continueplay) ,nl,nl,
186           if_then_else(Puzzle=1, write('(2) -> Randomly generate a problem to be
          solved.'), continueplay) ,nl,nl,
187           if_then_else(Puzzle=2, write('(1) -> See Solution(s) of the Original
          Game.'), continueplay) ,nl,nl,
188           if_then_else(Puzzle=2, write('(2) -> Randomly generate a problem to be
          solved.'), continueplay) ,nl,nl,
189           write('Your Option (Select 1 or 2): '), nl, read(Puzzle_Option),nl,nl,
```

```
190         if_then_else((Puzzle=1,Puzzle_Option=1),
            playProblem1(Solution,Pentagons1,5,0), continueplay) ,nl,
191         if_then_else((Puzzle=1,Puzzle_Option=2), (write('How many Colors?: (Minimum
            is 5 and Maximum is 12)  '), nl,read(NrColors)), continueplay) ,nl,nl,
192         if_then_else((Puzzle=2,Puzzle_Option=2), (write('How many Figures?: (Minimum
            is 3 and Maximum is 5)  '), nl,read(NrFigures)), continueplay) ,nl,nl,
193         if_then_else((Puzzle=1,Puzzle_Option=2,(NrColors<5; NrColors>12)), error,
            continueplay) ,nl,nl,
194         if_then_else((Puzzle=2,Puzzle_Option=2,(NrFigures<3; NrFigures>5)), error,
            continueplay) ,nl,nl,
195
196         if_then_else((Puzzle=1,Puzzle_Option=2),
            dodekduo_Puzzle1_Gen(Solution,1,NrColors), continueplay) ,nl,nl,
197         if_then_else((Puzzle=2,Puzzle_Option=2),
            dodekduo_Puzzle2_Gen(Solution,1,NrFigures), continueplay),
198
199         if_then_else((Puzzle=2,Puzzle_Option=1),
            playProblem2(Solution,Pentagons2,3,0), continueplay).
200
```

```
dodekduo_Puzzle1_Sol(Solution,Pentagons,NrColors,Input):-

%1-Laranja 2-Azul 3-Verde 4-Amarelo, 5-Rosa
length(Pentagons,Tam),
generateAllPentagons(Pentagons,Tam,[],AllPenta,0),

%Creating All Faces. According to the figure this is read from top to bottom and
left to right

Face1=[A1,A2,A3,A4,A5],
Face2=[B1,B2,B3,B4,B5],
Face3=[C1,C2,C3,C4,C5],
Face4=[D1,D2,D3,D4,D5],
Face5=[E1,E2,E3,E4,E5],
Face6=[F1,F2,F3,F4,F5],
Face7=[G1,G2,G3,G4,G5],
Face8=[H1,H2,H3,H4,H5],
Face9=[I1,I2,I3,I4,I5],
Face10=[J1,J2,J3,J4,J5],
Face11=[L1,L2,L3,L4,L5],
Face12=[M1,M2,M3,M4,M5],

table([Face1],AllPenta),
table([Face2],AllPenta),
table([Face3],AllPenta),
table([Face4],AllPenta),
table([Face5],AllPenta),
table([Face6],AllPenta),
table([Face7],AllPenta),
table([Face8],AllPenta),
table([Face9],AllPenta),
table([Face10],AllPenta),
table([Face11],AllPenta),
table([Face12],AllPenta),

%Two faces that are adjacent must have the same color



        %Nota: Pelo facto do Dodecaedro ter 60 simetrias de rotação,
        %esta restrição faz com que a solução corresponda à do enunciado.
        if_then_else(Input=0, (A1#=1 , A2#=2 , A3#=3 , A4#=4 , A5#=5), continueplay),


A1#=B1 , A2#=F1 , A3#=E1 , A4#=D1 , A5#=C1 ,

B1#=A1 , B2#=C5 , B3#=H5 , B4#=G1 , B5#=F2 ,

C1#=A5 , C2#=D5 , C3#=I5 , C4#=H1 , C5#=B2 ,

D1#=A4 , D2#=E5 , D3#=J5 , D4#=I1 , D5#=C2 ,

E1#=A3 , E2#=F5 , E3#=L5 , E4#=J1 , E5#=D2 ,

F1#=A2 , F2#=B5 , F3#=G5 , F4#=L1 , F5#=E2 ,

G1#=B4 , G2#=H4 , G3#=M5 , G4#=L2 , G5#=F3 ,

H1#=C4 , H2#=I4 , H3#=M1 , H4#=G2 , H5#=B3 ,

I1#=D4 , I2#=J4 , I3#=M2 , I4#=H2 , I5#=C3 ,

J1#=E4 , J2#=L4 , J3#=M3 , J4#=I2 , J5#=D3 ,

L1#=F4 , L2#=G4 , L3#=M4 , L4#=J2 , L5#=E3 ,

M1#=H3 , M2#=I3 , M3#=J3 , M4#=L3 , M5#=G3,

%Applying domain from 1 to 5 refering to the five colors that compose a face

domain(Face1,1,NrColors),
domain(Face2,1,NrColors),
```

```
72   domain(Face3,1,NrColors),
73   domain(Face4,1,NrColors),
74   domain(Face5,1,NrColors),
75   domain(Face6,1,NrColors),
76   domain(Face7,1,NrColors),
77   domain(Face8,1,NrColors),
78   domain(Face9,1,NrColors),
79   domain(Face10,1,NrColors),
80   domain(Face11,1,NrColors),
81   domain(Face12,1,NrColors),
82
83   %Check if all the triangles from a face have different colors
84
85   all_different(Face1),
86   all_different(Face2),
87   all_different(Face3),
88   all_different(Face4),
89   all_different(Face5),
90   all_different(Face6),
91   all_different(Face7),
92   all_different(Face8),
93   all_different(Face9),
94   all_different(Face10),
95   all_different(Face11),
96   all_different(Face12),
97
98
99   %Check if faces are different, including Rotations
100
101
        AllFaces=[Face1,Face2,Face3,Face4,Face5,Face6,Face7,Face8,Face9,Face10,Face11,Face
        12],
102     generateAllPentagons(AllFaces,12,[],AllRots,0),
103     allListsDifferent(AllRots),
104
105  write('**********Problem Input: 12 Pentagons**********'), nl,nl,
106  write('-> Nr of Colors: '), write(NrColors),nl,
107  printProblem(Pentagons), nl , nl,
108
109  append([Face1,Face2,Face3,Face4,Face5,Face6,Face7,Face8,Face9,Face10,Face11,Face12],So
     lution),
110  labeling([], Solution),
111
112  write('-> Note: Because there are 60 rotational symmetries '),nl,
113  write('in a regular dodecahedron there are (x solution * 60) total solutions to this
     problem.'),nl,nl,
114  write('-> The solution presented corresponds to the problem statement.'),nl,nl,
115
116  write('**********Solution:**********'), nl,nl,
117  printSolution(Pentagons,Solution),
118
119  nl,write('-> See other Solution? (Type 1 for YES and 2 for NO)'),nl,
120  write('Your Option (Select 1 or 2): '), nl, read(Option1),nl,nl,
121
122  if_then_else((Input=1,Option1=1),fail, continueplay),
123  if_then_else((Input=0,Option1=1),fail, continueplay),
124  if_then_else((Input=0,Option1>1),menu, continueplay).
```

```
dodekduo_Puzzle1_Gen(Solution,MinColor,MaxColor):-

%1-Laranja 2-Azul 3-Verde 4-Amarelo, 5-Rosa

PentaA=[PentaA1,PentaA2,PentaA3,PentaA4,PentaA5],
PentaB=[PentaB1,PentaB2,PentaB3,PentaB4,PentaB5],
PentaC=[PentaC1,PentaC2,PentaC3,PentaC4,PentaC5],
PentaD=[PentaD1,PentaD2,PentaD3,PentaD4,PentaD5],
PentaE=[PentaE1,PentaE2,PentaE3,PentaE4,PentaE5],
PentaF=[PentaF1,PentaF2,PentaF3,PentaF4,PentaF5],
PentaG=[PentaG1,PentaG2,PentaG3,PentaG4,PentaG5],
PentaH=[PentaH1,PentaH2,PentaH3,PentaH4,PentaH5],
PentaI=[PentaI1,PentaI2,PentaI3,PentaI4,PentaI5],
PentaJ=[PentaJ1,PentaJ2,PentaJ3,PentaJ4,PentaJ5],
PentaL=[PentaL1,PentaL2,PentaL3,PentaL4,PentaL5],
PentaM=[PentaM1,PentaM2,PentaM3,PentaM4,PentaM5],

domain(PentaA,MinColor,MaxColor),
domain(PentaB,MinColor,MaxColor),
domain(PentaC,MinColor,MaxColor),
domain(PentaD,MinColor,MaxColor),
domain(PentaE,MinColor,MaxColor),
domain(PentaF,MinColor,MaxColor),
domain(PentaG,MinColor,MaxColor),
domain(PentaH,MinColor,MaxColor),
domain(PentaI,MinColor,MaxColor),
domain(PentaJ,MinColor,MaxColor),
domain(PentaL,MinColor,MaxColor),
domain(PentaM,MinColor,MaxColor),

all_different(PentaA),
all_different(PentaB),
all_different(PentaC),
all_different(PentaD),
all_different(PentaE),
all_different(PentaF),
all_different(PentaG),
all_different(PentaH),
all_different(PentaI),
all_different(PentaJ),
all_different(PentaL),
all_different(PentaM),

Pentagons=[PentaA,PentaB,PentaC,PentaD,PentaE,PentaF,PentaG,PentaH,PentaI,PentaJ,PentaL,PentaM],
generateAllPentagons(Pentagons,12,[],AllPenta,0),
allListsDifferent(AllPenta),

fd_dom(PentaA1,Penta_A1),fd_dom(PentaA2,Penta_A2),fd_dom(PentaA3,Penta_A3),fd_dom(PentaA4,Penta_A4),fd_dom(PentaA5,Penta_A5),
fd_dom(PentaB1,Penta_B1),fd_dom(PentaB2,Penta_B2),fd_dom(PentaB3,Penta_B3),fd_dom(PentaB4,Penta_B4),fd_dom(PentaB5,Penta_B5),
fd_dom(PentaC1,Penta_C1),fd_dom(PentaC2,Penta_C2),fd_dom(PentaC3,Penta_C3),fd_dom(PentaC4,Penta_C4),fd_dom(PentaC5,Penta_C5),
fd_dom(PentaD1,Penta_D1),fd_dom(PentaD2,Penta_D2),fd_dom(PentaD3,Penta_D3),fd_dom(PentaD4,Penta_D4),fd_dom(PentaD5,Penta_D5),
fd_dom(PentaE1,Penta_E1),fd_dom(PentaE2,Penta_E2),fd_dom(PentaE3,Penta_E3),fd_dom(PentaE4,Penta_E4),fd_dom(PentaE5,Penta_E5),
fd_dom(PentaF1,Penta_F1),fd_dom(PentaF2,Penta_F2),fd_dom(PentaF3,Penta_F3),fd_dom(PentaF4,Penta_F4),fd_dom(PentaF5,Penta_F5),
fd_dom(PentaG1,Penta_G1),fd_dom(PentaG2,Penta_G2),fd_dom(PentaG3,Penta_G3),fd_dom(PentaG4,Penta_G4),fd_dom(PentaG5,Penta_G5),
fd_dom(PentaH1,Penta_H1),fd_dom(PentaH2,Penta_H2),fd_dom(PentaH3,Penta_H3),fd_dom(PentaH4,Penta_H4),fd_dom(PentaH5,Penta_H5),
fd_dom(PentaI1,Penta_I1),fd_dom(PentaI2,Penta_I2),fd_dom(PentaI3,Penta_I3),fd_dom(PentaI4,Penta_I4),fd_dom(PentaI5,Penta_I5),
fd_dom(PentaJ1,Penta_J1),fd_dom(PentaJ2,Penta_J2),fd_dom(PentaJ3,Penta_J3),fd_dom(PentaJ4,Penta_J4),fd_dom(PentaJ5,Penta_J5),
fd_dom(PentaL1,Penta_L1),fd_dom(PentaL2,Penta_L2),fd_dom(PentaL3,Penta_L3),fd_dom(PentaL4,Penta_L4),fd_dom(PentaL5,Penta_L5),
fd_dom(PentaM1,Penta_M1),fd_dom(PentaM2,Penta_M2),fd_dom(PentaM3,Penta_M3),fd_dom(PentaM4,Penta_M4),fd_dom(PentaM5,Penta_M5),
```

```
60
61  Penta_A=[Penta_A1,Penta_A2,Penta_A3,Penta_A4,Penta_A5],
62  Penta_B=[Penta_B1,Penta_B2,Penta_B3,Penta_B4,Penta_B5],
63  Penta_C=[Penta_C1,Penta_C2,Penta_C3,Penta_C4,Penta_C5],
64  Penta_D=[Penta_D1,Penta_D2,Penta_D3,Penta_D4,Penta_D5],
65  Penta_E=[Penta_E1,Penta_E2,Penta_E3,Penta_E4,Penta_E5],
66  Penta_F=[Penta_F1,Penta_F2,Penta_F3,Penta_F4,Penta_F5],
67  Penta_G=[Penta_G1,Penta_G2,Penta_G3,Penta_G4,Penta_G5],
68  Penta_H=[Penta_H1,Penta_H2,Penta_H3,Penta_H4,Penta_H5],
69  Penta_I=[Penta_I1,Penta_I2,Penta_I3,Penta_I4,Penta_I5],
70  Penta_J=[Penta_J1,Penta_J2,Penta_J3,Penta_J4,Penta_J5],
71  Penta_L=[Penta_L1,Penta_L2,Penta_L3,Penta_L4,Penta_L5],
72  Penta_M=[Penta_M1,Penta_M2,Penta_M3,Penta_M4,Penta_M5],
73
74  Pentagons_VAR=[Penta_A,Penta_B,Penta_C,Penta_D,Penta_E,Penta_F,Penta_G,Penta_H,Penta_I
    ,Penta_J,Penta_L,Penta_M],
75  generateAllPentagons(Pentagons_VAR,12,[],AllPenta_VAR,0),
76
77  %Creating All Faces. According to the figure this is read from top to bottom and
    left to right
78
79  Face1=[A1,A2,A3,A4,A5],
80  Face2=[B1,B2,B3,B4,B5],
81  Face3=[C1,C2,C3,C4,C5],
82  Face4=[D1,D2,D3,D4,D5],
83  Face5=[E1,E2,E3,E4,E5],
84  Face6=[F1,F2,F3,F4,F5],
85  Face7=[G1,G2,G3,G4,G5],
86  Face8=[H1,H2,H3,H4,H5],
87  Face9=[I1,I2,I3,I4,I5],
88  Face10=[J1,J2,J3,J4,J5],
89  Face11=[L1,L2,L3,L4,L5],
90  Face12=[M1,M2,M3,M4,M5],
91
92  %Applying domain from 1 to 5 refering to the five colors that compose a face
93
94  domain(Face1,MinColor,MaxColor),
95  domain(Face2,MinColor,MaxColor),
96  domain(Face3,MinColor,MaxColor),
97  domain(Face4,MinColor,MaxColor),
98  domain(Face5,MinColor,MaxColor),
99  domain(Face6,MinColor,MaxColor),
100 domain(Face7,MinColor,MaxColor),
101 domain(Face8,MinColor,MaxColor),
102 domain(Face9,MinColor,MaxColor),
103 domain(Face10,MinColor,MaxColor),
104 domain(Face11,MinColor,MaxColor),
105 domain(Face12,MinColor,MaxColor),
106
107 table([Face1],Pentagons_VAR),
108 table([Face2],Pentagons_VAR),
109 table([Face3],Pentagons_VAR),
110 table([Face4],Pentagons_VAR),
111 table([Face5],Pentagons_VAR),
112 table([Face7],Pentagons_VAR),
113 table([Face8],Pentagons_VAR),
114 table([Face9],Pentagons_VAR),
115 table([Face10],Pentagons_VAR),
116 table([Face11],Pentagons_VAR),
117 table([Face12],Pentagons_VAR),
118
119 fd_dom(A1,Face_A1),fd_dom(A2,Face_A2),fd_dom(A3,Face_A3),fd_dom(A4,Face_A4),fd_dom(A5,
    Face_A5),
120 fd_dom(B1,Face_B1),fd_dom(B2,Face_B2),fd_dom(B3,Face_B3),fd_dom(B4,Face_B4),fd_dom(B5,
    Face_B5),
121 fd_dom(C1,Face_C1),fd_dom(C2,Face_C2),fd_dom(C3,Face_C3),fd_dom(C4,Face_C4),fd_dom(C5,
    Face_C5),
122 fd_dom(D1,Face_D1),fd_dom(D2,Face_D2),fd_dom(D3,Face_D3),fd_dom(D4,Face_D4),fd_dom(D5,
    Face_D5),
123 fd_dom(E1,Face_E1),fd_dom(E2,Face_E2),fd_dom(E3,Face_E3),fd_dom(E4,Face_E4),fd_dom(E5,
    Face_E5),
124 fd_dom(F1,Face_F1),fd_dom(F2,Face_F2),fd_dom(F3,Face_F3),fd_dom(F4,Face_F4),fd_dom(F5,
```

```
    Face_F5),
125 fd_dom(G1,Face_G1),fd_dom(G2,Face_G2),fd_dom(G3,Face_G3),fd_dom(G4,Face_G4),fd_dom(G5,
    Face_G5),
126 fd_dom(H1,Face_H1),fd_dom(H2,Face_H2),fd_dom(H3,Face_H3),fd_dom(H4,Face_H4),fd_dom(H5,
    Face_H5),
127 fd_dom(I1,Face_I1),fd_dom(I2,Face_I2),fd_dom(I3,Face_I3),fd_dom(I4,Face_I4),fd_dom(I5,
    Face_I5),
128 fd_dom(J1,Face_J1),fd_dom(J2,Face_J2),fd_dom(J3,Face_J3),fd_dom(J4,Face_J4),fd_dom(J5,
    Face_J5),
129 fd_dom(L1,Face_L1),fd_dom(L2,Face_L2),fd_dom(L3,Face_L3),fd_dom(L4,Face_L4),fd_dom(L5,
    Face_L5),
130 fd_dom(M1,Face_M1),fd_dom(M2,Face_M2),fd_dom(M3,Face_M3),fd_dom(M4,Face_M4),fd_dom(M5,
    Face_M5),
131
132 Face_A=[Face_A1,Face_A2,Face_A3,Face_A4,Face_A5],
133 Face_B=[Face_B1,Face_B2,Face_B3,Face_B4,Face_B5],
134 Face_C=[Face_C1,Face_C2,Face_C3,Face_C4,Face_C5],
135 Face_D=[Face_D1,Face_D2,Face_D3,Face_D4,Face_D5],
136 Face_E=[Face_E1,Face_E2,Face_E3,Face_E4,Face_E5],
137 Face_F=[Face_F1,Face_F2,Face_F3,Face_F4,Face_F5],
138 Face_G=[Face_G1,Face_G2,Face_G3,Face_G4,Face_G5],
139 Face_H=[Face_H1,Face_H2,Face_H3,Face_H4,Face_H5],
140 Face_I=[Face_I1,Face_I2,Face_I3,Face_I4,Face_I5],
141 Face_J=[Face_J1,Face_J2,Face_J3,Face_J4,Face_J5],
142 Face_L=[Face_L1,Face_L2,Face_L3,Face_L4,Face_L5],
143 Face_M=[Face_M1,Face_M2,Face_M3,Face_M4,Face_M5],
144
145 Faces_VAR=[Face_A,Face_B,Face_C,Face_D,Face_E,Face_F,Face_G,Face_H,Face_I,Face_J,Face_
    L,Face_M],
146 generateAllPentagons(Faces_VAR,12,[],AllFaces_VAR,0),
147
148
149
150 table([PentaA],AllFaces_VAR),
151 table([PentaB],AllFaces_VAR),
152 table([PentaC],AllFaces_VAR),
153 table([PentaD],AllFaces_VAR),
154 table([PentaE],AllFaces_VAR),
155 table([PentaF],AllFaces_VAR),
156 table([PentaG],AllFaces_VAR),
157 table([PentaH],AllFaces_VAR),
158 table([PentaI],AllFaces_VAR),
159 table([PentaJ],AllFaces_VAR),
160 table([PentaL],AllFaces_VAR),
161 table([PentaM],AllFaces_VAR),
162
163 %Two faces that are adjacent must have the same color
164
165 A1#=B1 #/\ A2#=F1 #/\ A3#=E1 #/\ A4#=D1 #/\ A5#=C1 #/\
166
167 B1#=A1 #/\ B2#=C5 #/\ B3#=H5 #/\ B4#=G1 #/\ B5#=F2 #/\
168
169 C1#=A5 #/\ C2#=D5 #/\ C3#=I5 #/\ C4#=H1 #/\ C5#=B2 #/\
170
171 D1#=A4 #/\ D2#=E5 #/\ D3#=J5 #/\ D4#=I1 #/\ D5#=C2 #/\
172
173 E1#=A3 #/\ E2#=F5 #/\ E3#=L5 #/\ E4#=J1 #/\ E5#=D2 #/\
174
175 F1#=A2 #/\ F2#=B5 #/\ F3#=G5 #/\ F4#=L1 #/\ F5#=E2 #/\
176
177 G1#=B4 #/\ G2#=H4 #/\ G3#=M5 #/\ G4#=L2 #/\ G5#=F3 #/\
178
179 H1#=C4 #/\ H2#=I4 #/\ H3#=M1 #/\ H4#=G2 #/\ H5#=B3 #/\
180
181 I1#=D4 #/\ I2#=J4 #/\ I3#=M2 #/\ I4#=H2 #/\ I5#=C3 #/\
182
183 J1#=E4 #/\ J2#=L4 #/\ J3#=M3 #/\ J4#=I2 #/\ J5#=D3 #/\
184
185 L1#=F4 #/\ L2#=G4 #/\ L3#=M4 #/\ L4#=J2 #/\ L5#=E3 #/\
186
187 M1#=H3 #/\ M2#=I3 #/\ M3#=J3 #/\ M4#=L3 #/\ M5#=G3,
188
```

```
189   %Check if all the triangles from a face have different colors
190
191   all_different(Face1),
192   all_different(Face2),
193   all_different(Face3),
194   all_different(Face4),
195   all_different(Face5),
196   all_different(Face6),
197   all_different(Face7),
198   all_different(Face8),
199   all_different(Face9),
200   all_different(Face10),
201   all_different(Face11),
202   all_different(Face12),
203
204
205
206   %Check if faces are different, including Rotations
207
208   AllFaces=[Face1,Face2,Face3,Face4,Face5,Face6,Face7,Face8,Face9,Face10,Face11,Face12],
209
210   append(AllFaces,AllElements),
211   MaxColor2 is MaxColor+1,
212   forceColors(AllElements,MaxColor2,1),
213
214   generateAllPentagons(AllFaces,12,[],AllRots,0),
215
216   allListsDifferent(AllRots),
217
218   append([Face1,Face2,Face3,Face4,Face5,Face6,Face7,Face8,Face9,Face10,Face11,Face12],So
      lution),
219   labeling([], Solution),
220
221   generateProb(AllFaces,Problem),
222
223   if_then_else(dodekduo_Puzzle1_Sol(NewSolution,Problem,MaxColor,1),continueplay,(nl,nl,
      write('No More Solutions!!!'),nl,nl)),
224
225   write('-> Generate other Problem? (Type 1 for YES and 2 for NO)'),nl,
226   write('Your Option (Select 1 or 2): '), nl, read(Option1),nl,nl,
227
228   if_then_else(Option1=1,fail, menu).
```

```
dodekduo_Puzzle2_Sol(Solution,Pentagons,NrFigures,Input):-


%1-Retangulo 2-Semi-Circulo 3-Triangulo

length(Pentagons,Tam),
generateAllPentagons(Pentagons,Tam,[],AllPenta,0),

%Creating All Faces. According to the figure this is read from top to bottom and
left to right

Face1=[A1,A2,A3,A4,A5],
Face2=[B1,B2,B3,B4,B5],
Face3=[C1,C2,C3,C4,C5],
Face4=[D1,D2,D3,D4,D5],
Face5=[E1,E2,E3,E4,E5],
Face6=[F1,F2,F3,F4,F5],
Face7=[G1,G2,G3,G4,G5],
Face8=[H1,H2,H3,H4,H5],
Face9=[I1,I2,I3,I4,I5],
Face10=[J1,J2,J3,J4,J5],
Face11=[L1,L2,L3,L4,L5],
Face12=[M1,M2,M3,M4,M5],

table([Face1],AllPenta),
table([Face2],AllPenta),
table([Face3],AllPenta),
table([Face4],AllPenta),
table([Face5],AllPenta),
table([Face6],AllPenta),
table([Face7],AllPenta),
table([Face8],AllPenta),
table([Face9],AllPenta),
table([Face10],AllPenta),
table([Face11],AllPenta),
table([Face12],AllPenta),

%Two faces that are adjacent must have the same color

            %Nota: Pelo facto do Dodecaedro ter 60 simetrias de rotação,
            %esta restrição faz com que as soluções correspondam à do enunciado.
            if_then_else(Input=0,(A1#=1 , A2#=2 , A3#=3 , A4#=3 ,
            A5#=1),continueplay),

A1#=B1 , A2#=F1 , A3#=E1 , A4#=D1 , A5#=C1 ,

B1#=A1 , B2#=C5 , B3#=H5 , B4#=G1 , B5#=F2 ,

C1#=A5 , C2#=D5 , C3#=I5 , C4#=H1 , C5#=B2 ,

D1#=A4 , D2#=E5 , D3#=J5 , D4#=I1 , D5#=C2 ,

E1#=A3 , E2#=F5 , E3#=L5 , E4#=J1 , E5#=D2 ,

F1#=A2 , F2#=B5 , F3#=G5 , F4#=L1 , F5#=E2 ,

G1#=B4 , G2#=H4 , G3#=M5 , G4#=L2 , G5#=F3 ,

H1#=C4 , H2#=I4 , H3#=M1 , H4#=G2 , H5#=B3 ,

I1#=D4 , I2#=J4 , I3#=M2 , I4#=H2 , I5#=C3 ,

J1#=E4 , J2#=L4 , J3#=M3 , J4#=I2 , J5#=D3 ,

L1#=F4 , L2#=G4 , L3#=M4 , L4#=J2 , L5#=E3 ,

M1#=H3 , M2#=I3 , M3#=J3 , M4#=L3 , M5#=G3,

%Applying domain from 1 to 5 refering to the five colors that compose a face

domain(Face1,1,NrFigures),
domain(Face2,1,NrFigures),
```

```
71   domain(Face3,1,NrFigures),
72   domain(Face4,1,NrFigures),
73   domain(Face5,1,NrFigures),
74   domain(Face6,1,NrFigures),
75   domain(Face7,1,NrFigures),
76   domain(Face8,1,NrFigures),
77   domain(Face9,1,NrFigures),
78   domain(Face10,1,NrFigures),
79   domain(Face11,1,NrFigures),
80   domain(Face12,1,NrFigures),
81
82   %Check if faces are different, including Rotations
83
84
       AllFaces=[Face1,Face2,Face3,Face4,Face5,Face6,Face7,Face8,Face9,Face10,Face11,Face
       12],
85     generateAllPentagons(AllFaces,12,[],AllRots,0),
86     allListsDifferent(AllRots),
87
88   write('**********Problem Input: 12 Pentagons**********'), nl,nl,
89   write('-> Nr of Figures: '), write(NrFigures),nl,
90   printProblem(Pentagons), nl , nl,
91
92   append([Face1,Face2,Face3,Face4,Face5,Face6,Face7,Face8,Face9,Face10,Face11,Face12],So
     lution),
93   labeling([], Solution),
94
95   write('-> Note: Because there are 60 rotational symmetries '),nl,
96   write('in a regular dodecahedron there are (x solution * 60) total solutions to this
     problem.'),nl,nl,
97   write('-> The solution presented corresponds to the problem statement.'),nl,nl,
98
99   write('**********Solution:**********'), nl,nl,
100  printSolution(Pentagons,Solution),
101
102  nl,write('-> See other Solution? (Type 1 for YES and 2 for NO)'),nl,
103  write('Your Option (Select 1 or 2): '), nl, read(Option1),nl,nl,
104
105  if_then_else((Input=1,Option1=1),fail, continueplay),
106  if_then_else((Input=0,Option1=1),fail, continueplay),
107  if_then_else((Input=0,Option1>1),menu, continueplay).
```

```
dodekduo_Puzzle2_Gen(Solution,MinFigures,MaxFigures):-

%1-Retangulo 2-Semi-Circulo 3-Triangulo

PentaA=[PentaA1,PentaA2,PentaA3,PentaA4,PentaA5],
PentaB=[PentaB1,PentaB2,PentaB3,PentaB4,PentaB5],
PentaC=[PentaC1,PentaC2,PentaC3,PentaC4,PentaC5],
PentaD=[PentaD1,PentaD2,PentaD3,PentaD4,PentaD5],
PentaE=[PentaE1,PentaE2,PentaE3,PentaE4,PentaE5],
PentaF=[PentaF1,PentaF2,PentaF3,PentaF4,PentaF5],
PentaG=[PentaG1,PentaG2,PentaG3,PentaG4,PentaG5],
PentaH=[PentaH1,PentaH2,PentaH3,PentaH4,PentaH5],
PentaI=[PentaI1,PentaI2,PentaI3,PentaI4,PentaI5],
PentaJ=[PentaJ1,PentaJ2,PentaJ3,PentaJ4,PentaJ5],
PentaL=[PentaL1,PentaL2,PentaL3,PentaL4,PentaL5],
PentaM=[PentaM1,PentaM2,PentaM3,PentaM4,PentaM5],

domain(PentaA,MinFigures,MaxFigures),
domain(PentaB,MinFigures,MaxFigures),
domain(PentaC,MinFigures,MaxFigures),
domain(PentaD,MinFigures,MaxFigures),
domain(PentaE,MinFigures,MaxFigures),
domain(PentaF,MinFigures,MaxFigures),
domain(PentaG,MinFigures,MaxFigures),
domain(PentaH,MinFigures,MaxFigures),
domain(PentaI,MinFigures,MaxFigures),
domain(PentaJ,MinFigures,MaxFigures),
domain(PentaL,MinFigures,MaxFigures),
domain(PentaM,MinFigures,MaxFigures),

Pentagons=[PentaA,PentaB,PentaC,PentaD,PentaE,PentaF,PentaG,PentaH,PentaI,PentaJ,PentaL,PentaM],
generateAllPentagons(Pentagons,12,[],AllPenta,0),
allListsDifferent(AllPenta),

fd_dom(PentaA1,Penta_A1),fd_dom(PentaA2,Penta_A2),fd_dom(PentaA3,Penta_A3),fd_dom(PentaA4,Penta_A4),fd_dom(PentaA5,Penta_A5),
fd_dom(PentaB1,Penta_B1),fd_dom(PentaB2,Penta_B2),fd_dom(PentaB3,Penta_B3),fd_dom(PentaB4,Penta_B4),fd_dom(PentaB5,Penta_B5),
fd_dom(PentaC1,Penta_C1),fd_dom(PentaC2,Penta_C2),fd_dom(PentaC3,Penta_C3),fd_dom(PentaC4,Penta_C4),fd_dom(PentaC5,Penta_C5),
fd_dom(PentaD1,Penta_D1),fd_dom(PentaD2,Penta_D2),fd_dom(PentaD3,Penta_D3),fd_dom(PentaD4,Penta_D4),fd_dom(PentaD5,Penta_D5),
fd_dom(PentaE1,Penta_E1),fd_dom(PentaE2,Penta_E2),fd_dom(PentaE3,Penta_E3),fd_dom(PentaE4,Penta_E4),fd_dom(PentaE5,Penta_E5),
fd_dom(PentaF1,Penta_F1),fd_dom(PentaF2,Penta_F2),fd_dom(PentaF3,Penta_F3),fd_dom(PentaF4,Penta_F4),fd_dom(PentaF5,Penta_F5),
fd_dom(PentaG1,Penta_G1),fd_dom(PentaG2,Penta_G2),fd_dom(PentaG3,Penta_G3),fd_dom(PentaG4,Penta_G4),fd_dom(PentaG5,Penta_G5),
fd_dom(PentaH1,Penta_H1),fd_dom(PentaH2,Penta_H2),fd_dom(PentaH3,Penta_H3),fd_dom(PentaH4,Penta_H4),fd_dom(PentaH5,Penta_H5),
fd_dom(PentaI1,Penta_I1),fd_dom(PentaI2,Penta_I2),fd_dom(PentaI3,Penta_I3),fd_dom(PentaI4,Penta_I4),fd_dom(PentaI5,Penta_I5),
fd_dom(PentaJ1,Penta_J1),fd_dom(PentaJ2,Penta_J2),fd_dom(PentaJ3,Penta_J3),fd_dom(PentaJ4,Penta_J4),fd_dom(PentaJ5,Penta_J5),
fd_dom(PentaL1,Penta_L1),fd_dom(PentaL2,Penta_L2),fd_dom(PentaL3,Penta_L3),fd_dom(PentaL4,Penta_L4),fd_dom(PentaL5,Penta_L5),
fd_dom(PentaM1,Penta_M1),fd_dom(PentaM2,Penta_M2),fd_dom(PentaM3,Penta_M3),fd_dom(PentaM4,Penta_M4),fd_dom(PentaM5,Penta_M5),

Penta_A=[Penta_A1,Penta_A2,Penta_A3,Penta_A4,Penta_A5],
Penta_B=[Penta_B1,Penta_B2,Penta_B3,Penta_B4,Penta_B5],
Penta_C=[Penta_C1,Penta_C2,Penta_C3,Penta_C4,Penta_C5],
Penta_D=[Penta_D1,Penta_D2,Penta_D3,Penta_D4,Penta_D5],
Penta_E=[Penta_E1,Penta_E2,Penta_E3,Penta_E4,Penta_E5],
Penta_F=[Penta_F1,Penta_F2,Penta_F3,Penta_F4,Penta_F5],
Penta_G=[Penta_G1,Penta_G2,Penta_G3,Penta_G4,Penta_G5],
Penta_H=[Penta_H1,Penta_H2,Penta_H3,Penta_H4,Penta_H5],
Penta_I=[Penta_I1,Penta_I2,Penta_I3,Penta_I4,Penta_I5],
Penta_J=[Penta_J1,Penta_J2,Penta_J3,Penta_J4,Penta_J5],
Penta_L=[Penta_L1,Penta_L2,Penta_L3,Penta_L4,Penta_L5],
Penta_M=[Penta_M1,Penta_M2,Penta_M3,Penta_M4,Penta_M5],
```

```
60
61  Pentagons_VAR=[Penta_A,Penta_B,Penta_C,Penta_D,Penta_E,Penta_F,Penta_G,Penta_H,Penta_I
    ,Penta_J,Penta_L,Penta_M],
62  generateAllPentagons(Pentagons_VAR,12,[],AllPenta_VAR,0),
63
64  %Creating All Faces. According to the figure this is read from top to bottom and
    left to right
65
66  Face1=[A1,A2,A3,A4,A5],
67  Face2=[B1,B2,B3,B4,B5],
68  Face3=[C1,C2,C3,C4,C5],
69  Face4=[D1,D2,D3,D4,D5],
70  Face5=[E1,E2,E3,E4,E5],
71  Face6=[F1,F2,F3,F4,F5],
72  Face7=[G1,G2,G3,G4,G5],
73  Face8=[H1,H2,H3,H4,H5],
74  Face9=[I1,I2,I3,I4,I5],
75  Face10=[J1,J2,J3,J4,J5],
76  Face11=[L1,L2,L3,L4,L5],
77  Face12=[M1,M2,M3,M4,M5],
78
79  %Applying domain from 1 to 5 refering to the five colors that compose a face
80
81  domain(Face1,MinFigures,MaxFigures),
82  domain(Face2,MinFigures,MaxFigures),
83  domain(Face3,MinFigures,MaxFigures),
84  domain(Face4,MinFigures,MaxFigures),
85  domain(Face5,MinFigures,MaxFigures),
86  domain(Face6,MinFigures,MaxFigures),
87  domain(Face7,MinFigures,MaxFigures),
88  domain(Face8,MinFigures,MaxFigures),
89  domain(Face9,MinFigures,MaxFigures),
90  domain(Face10,MinFigures,MaxFigures),
91  domain(Face11,MinFigures,MaxFigures),
92  domain(Face12,MinFigures,MaxFigures),
93
94  table([Face1],Pentagons_VAR),
95  table([Face2],Pentagons_VAR),
96  table([Face3],Pentagons_VAR),
97  table([Face4],Pentagons_VAR),
98  table([Face5],Pentagons_VAR),
99  table([Face7],Pentagons_VAR),
100 table([Face8],Pentagons_VAR),
101 table([Face9],Pentagons_VAR),
102 table([Face10],Pentagons_VAR),
103 table([Face11],Pentagons_VAR),
104 table([Face12],Pentagons_VAR),
105
106 fd_dom(A1,Face_A1),fd_dom(A2,Face_A2),fd_dom(A3,Face_A3),fd_dom(A4,Face_A4),fd_dom(A5,
    Face_A5),
107 fd_dom(B1,Face_B1),fd_dom(B2,Face_B2),fd_dom(B3,Face_B3),fd_dom(B4,Face_B4),fd_dom(B5,
    Face_B5),
108 fd_dom(C1,Face_C1),fd_dom(C2,Face_C2),fd_dom(C3,Face_C3),fd_dom(C4,Face_C4),fd_dom(C5,
    Face_C5),
109 fd_dom(D1,Face_D1),fd_dom(D2,Face_D2),fd_dom(D3,Face_D3),fd_dom(D4,Face_D4),fd_dom(D5,
    Face_D5),
110 fd_dom(E1,Face_E1),fd_dom(E2,Face_E2),fd_dom(E3,Face_E3),fd_dom(E4,Face_E4),fd_dom(E5,
    Face_E5),
111 fd_dom(F1,Face_F1),fd_dom(F2,Face_F2),fd_dom(F3,Face_F3),fd_dom(F4,Face_F4),fd_dom(F5,
    Face_F5),
112 fd_dom(G1,Face_G1),fd_dom(G2,Face_G2),fd_dom(G3,Face_G3),fd_dom(G4,Face_G4),fd_dom(G5,
    Face_G5),
113 fd_dom(H1,Face_H1),fd_dom(H2,Face_H2),fd_dom(H3,Face_H3),fd_dom(H4,Face_H4),fd_dom(H5,
    Face_H5),
114 fd_dom(I1,Face_I1),fd_dom(I2,Face_I2),fd_dom(I3,Face_I3),fd_dom(I4,Face_I4),fd_dom(I5,
    Face_I5),
115 fd_dom(J1,Face_J1),fd_dom(J2,Face_J2),fd_dom(J3,Face_J3),fd_dom(J4,Face_J4),fd_dom(J5,
    Face_J5),
116 fd_dom(L1,Face_L1),fd_dom(L2,Face_L2),fd_dom(L3,Face_L3),fd_dom(L4,Face_L4),fd_dom(L5,
    Face_L5),
117 fd_dom(M1,Face_M1),fd_dom(M2,Face_M2),fd_dom(M3,Face_M3),fd_dom(M4,Face_M4),fd_dom(M5,
    Face_M5),
```

```
118
119   Face_A=[Face_A1,Face_A2,Face_A3,Face_A4,Face_A5],
120   Face_B=[Face_B1,Face_B2,Face_B3,Face_B4,Face_B5],
121   Face_C=[Face_C1,Face_C2,Face_C3,Face_C4,Face_C5],
122   Face_D=[Face_D1,Face_D2,Face_D3,Face_D4,Face_D5],
123   Face_E=[Face_E1,Face_E2,Face_E3,Face_E4,Face_E5],
124   Face_F=[Face_F1,Face_F2,Face_F3,Face_F4,Face_F5],
125   Face_G=[Face_G1,Face_G2,Face_G3,Face_G4,Face_G5],
126   Face_H=[Face_H1,Face_H2,Face_H3,Face_H4,Face_H5],
127   Face_I=[Face_I1,Face_I2,Face_I3,Face_I4,Face_I5],
128   Face_J=[Face_J1,Face_J2,Face_J3,Face_J4,Face_J5],
129   Face_L=[Face_L1,Face_L2,Face_L3,Face_L4,Face_L5],
130   Face_M=[Face_M1,Face_M2,Face_M3,Face_M4,Face_M5],
131
132   Faces_VAR=[Face_A,Face_B,Face_C,Face_D,Face_E,Face_F,Face_G,Face_H,Face_I,Face_J,Face_
      L,Face_M],
133   generateAllPentagons(Faces_VAR,12,[],AllFaces_VAR,0),
134
135
136   table([PentaA],AllFaces_VAR),
137   table([PentaB],AllFaces_VAR),
138   table([PentaC],AllFaces_VAR),
139   table([PentaD],AllFaces_VAR),
140   table([PentaE],AllFaces_VAR),
141   table([PentaF],AllFaces_VAR),
142   table([PentaG],AllFaces_VAR),
143   table([PentaH],AllFaces_VAR),
144   table([PentaI],AllFaces_VAR),
145   table([PentaJ],AllFaces_VAR),
146   table([PentaL],AllFaces_VAR),
147   table([PentaM],AllFaces_VAR),
148
149   %Two faces that are adjacent must have the same color
150
151   A1#=B1 #/\ A2#=F1 #/\ A3#=E1 #/\ A4#=D1 #/\ A5#=C1 #/\
152
153   B1#=A1 #/\ B2#=C5 #/\ B3#=H5 #/\ B4#=G1 #/\ B5#=F2 #/\
154
155   C1#=A5 #/\ C2#=D5 #/\ C3#=I5 #/\ C4#=H1 #/\ C5#=B2 #/\
156
157   D1#=A4 #/\ D2#=E5 #/\ D3#=J5 #/\ D4#=I1 #/\ D5#=C2 #/\
158
159   E1#=A3 #/\ E2#=F5 #/\ E3#=L5 #/\ E4#=J1 #/\ E5#=D2 #/\
160
161   F1#=A2 #/\ F2#=B5 #/\ F3#=G5 #/\ F4#=L1 #/\ F5#=E2 #/\
162
163   G1#=B4 #/\ G2#=H4 #/\ G3#=M5 #/\ G4#=L2 #/\ G5#=F3 #/\
164
165   H1#=C4 #/\ H2#=I4 #/\ H3#=M1 #/\ H4#=G2 #/\ H5#=B3 #/\
166
167   I1#=D4 #/\ I2#=J4 #/\ I3#=M2 #/\ I4#=H2 #/\ I5#=C3 #/\
168
169   J1#=E4 #/\ J2#=L4 #/\ J3#=M3 #/\ J4#=I2 #/\ J5#=D3 #/\
170
171   L1#=F4 #/\ L2#=G4 #/\ L3#=M4 #/\ L4#=J2 #/\ L5#=E3 #/\
172
173   M1#=H3 #/\ M2#=I3 #/\ M3#=J3 #/\ M4#=L3 #/\ M5#=G3,
174
175
176   %Check if faces are different, including Rotations
177
178   AllFaces=[Face1,Face2,Face3,Face4,Face5,Face6,Face7,Face8,Face9,Face10,Face11,Face12],
179
180   append(AllFaces,AllElements),
181   MaxFigures2 is MaxFigures+1,
182   forceShapes(AllFaces,MaxFigures2,1),
183
184   generateAllPentagons(AllFaces,12,[],AllRots,0),
185
186   allListsDifferent(AllRots),
187
188   append([Face1,Face2,Face3,Face4,Face5,Face6,Face7,Face8,Face9,Face10,Face11,Face12],So
```

```
    lution),
189 labeling([], Solution),
190
191 generateProb(AllFaces,Problem),
192
193 if_then_else(dodekduo_Puzzle2_Sol(NewSolution,Problem,MaxFigures,1),continueplay,(nl,n
    l,write('No More Solutions!!!'),nl,nl)),
194
195 write('-> Generate other Problem? (Type 1 for YES and 2 for NO)'),nl,
196 write('Your Option (Select 1 or 2): '), nl, read(Option1),nl,nl,
197
198 if_then_else(Option1=1,fail, menu).
```