

Documentación segundo laboratorio

Programación Lógica

| Integrantes | C.I. | Email |
|-----------------|-------------|---------------------------|
| Ian Arazny | 5.147.233-7 | ian.arazny@fing.edu.uy |
| Favio Cardoso | 5.140.801-7 | favio.cardoso@fing.edu.uy |
| Alejandro Wurm | 5.224.817-9 | alejandrowurm@gmail.com |
| Micaela Vaucher | 5.204.544-6 | micaela.vaucher@gmail.com |

Índice

| | |
|---|----------|
| Índice | 2 |
| ia_det: | 3 |
| puntaje(+Datos, +Cat,-Puntos) | 3 |
| puntaje_tablero(+Tablero, -Puntaje) | 3 |
| ajustar_tablero(+Tablero,+Categoria,+Puntaje,-TableroSalida). | 3 |
| cambio_datos(+Datos,+Tablero,+Estrategia,-Patron) | 3 |
| eleccion_slot(+Datos,+Tablero,+Estrategia,-Categoria) | 5 |
| ia_prob: | 7 |
| puntaje(+Datos, +Cat,-Puntos) | 7 |
| puntaje_tablero(+Tablero, -Puntaje) | 7 |
| ajustar_tablero(+Tablero,+Categoria,+Puntaje,-TableroSalida). | 7 |
| cambio_datos(+Datos,+Tablero,+Estrategia,-Patron) | 7 |
| eleccion_slot(+Datos,+Tablero,+Estrategia,-Categoria) | 8 |
| Función prob | 8 |
| Tabla comparativa | 8 |

ia_det:

puntaje(+Dados, +Cat,-Puntos)

Calcula el puntaje para categorías especiales basándose en permutaciones de los dados.

puntaje_tablero(+Tablero, -Puntaje)

puntaje_tablero([C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,C12,C13], Puntaje) :-
 suma([C1,C2,C3,C4,C5,C6], P1, 0),
 llevaBonus(P1, Bonus),
 suma([C7,C8,C9,C10,C11,C12,C13], P2, 0),
 Puntaje is P1 + Bonus + P2.

Calcula el puntaje total sumando los puntajes de todas las categorías y añadiendo el bonus si la suma de los puntajes de las categorías superiores es mayor o igual a 63.

Donde

- llevaBonus: determina si se obtiene el bonus de 35 puntos.
- suma: los valores de una lista de estructuras (Categoría, Puntaje) son sumados.

ajustar_tablero(+Tablero,+Categoría,+Puntaje,-TableroSalida).

ajustar_tablero([], _, _, []).
ajustar_tablero([s(C,_)|L], C, P, [s(C, P)|Ts]):-
 ajustar_tablero(L, C, P, Ts).
ajustar_tablero([s(C1,X)|L], C, P, [s(C1,X)|Ts]):-
 ajustar_tablero(L, C, P, Ts).

Este predicado recorre el tablero y actualiza el puntaje de la categoría C con el valor P.

cambio_dados(+Dados,+Tablero,+Estrategia,-Patron)

La función cambio_dados/4 implementa una estrategia para el juego de Yahtzee. Evalúa seis posibles configuraciones en orden de preferencia, priorizando combinaciones que resulten en mayores puntajes. A continuación, se describen cada una de estas posibilidades:

1. Yahtzee:

cambio_dados(Dados, Tablero, ia_det, [0,0,0,0,0]):-

```
puntaje(Dados, yahtzee, 50),  
ocupada_categoria(yahtzee, Tablero, 0).
```

En primer lugar, la función verifica si los dados forman un Yahtzee (todos los dados iguales) y si la categoría Yahtzee está disponible en el tablero. Si ambos criterios se cumplen, el patrón resultante es [0,0,0,0,0], indicando que no se cambia ningún dado.

2. Large Straight

```
cambio_datos(Dados, Tablero, ia_det, [0,0,0,0,0]):-  
    puntaje(Dados, large_straight, 40),  
    ocupada_categoria(large_straight, Tablero, 0).
```

Si no se puede formar un Yahtzee, la función comprueba si los dados forman una Large Straight (una secuencia consecutiva de cinco números) y si esta categoría está disponible. De nuevo, si ambos criterios se cumplen, no se cambian los dados ([0,0,0,0,0]).

3. Small Straight

```
cambio_datos(Dados, Tablero, ia_det, [0,0,0,0,0]):-  
    puntaje(Dados, small_straight, 30),  
    ocupada_categoria(small_straight, Tablero, 0).
```

Si no se puede formar una Large Straight, se verifica si los dados forman una Small Straight (una secuencia consecutiva de cuatro números) y si esta categoría está libre. Si es así, el patrón de cambio es [0,0,0,0,0].

4. Full House

```
cambio_datos(Dados, Tablero, ia_det, [0,0,0,0,0]):-  
    puntaje(Dados, full_house, 25),  
    ocupada_categoria(full_house, Tablero, 0).
```

Si no se logra ninguna de las combinaciones anteriores, la función comprueba si los dados forman un Full House (tres de un tipo y dos de otro) y si esta categoría está disponible en el tablero. De cumplirse ambos criterios, no se cambia ningún dado ([0,0,0,0,0]).

5. En caso de no tener repetidos ni escalera

```
cambio_datos(Dados, _, ia_det, [1,1,1,1,1]):-  
    sin_repetidos(Dados).
```

Si los dados no cumplen con ninguna de las combinaciones anteriores y además no tienen repeticiones (todos los números son diferentes), se decide cambiar todos los dados. El patrón resultante es [1,1,1,1,1].

6. Máximas Repeticiones

```
cambio_datos(Dados, _, ia_det, Patron):-  
    maximas_repeticiones(Dados, _, N),  
    reemplazar(N, Dados, Patron).
```

Finalmente, si ninguna de las condiciones anteriores se cumple, la función evalúa cuál es el número que más se repite en los dados (maximas_repeticiones/3). Todos

los dados que no coinciden con este número se marcan para ser cambiados mediante la función reemplazar/3, generando así el patrón de cambio Patron.

eleccion_slot(+Datos,+Tablero,+Estrategia,-Categoria)

La función `eleccion_slot/4` implementa una estrategia para seleccionar la mejor categoría disponible en el turno de una partida. Esta función tiene 9 posibilidades que se evalúan en el orden de preferencia definido. A continuación, se describen cada una de estas posibilidades:

1. Yahtzee

`eleccion_slot(Datos, Tablero, ia_det, yahtzee):-`

`puntaje(Datos, yahtzee, 50),`

`ocupada_categoria(yahtzee, Tablero, 0).`

En primer lugar, la función verifica si los dados forman un Yahtzee (todos los dados iguales) y si la categoría Yahtzee está disponible en el tablero. Si ambos criterios se cumplen, se elige la categoría yahtzee.

2. Large Straight

`eleccion_slot(Datos, Tablero, ia_det, large_straight):-`

`puntaje(Datos, large_straight, 40),`

`ocupada_categoria(large_straight, Tablero, 0).`

Si no se puede formar un Yahtzee, la función comprueba si los dados forman una Large Straight (una secuencia consecutiva de cinco números) y si esta categoría está disponible. Si ambos criterios se cumplen, se elige la categoría large_straight.

3. Small Straight

`eleccion_slot(Datos, Tablero, ia_det, small_straight):-`

`puntaje(Datos, small_straight, 30),`

`ocupada_categoria(small_straight, Tablero, 0).`

Si no se puede formar una Large Straight, se verifica si los dados forman una Small Straight (una secuencia consecutiva de cuatro números) y si esta categoría está libre. Si es así, se elige la categoría small_straight.

4. Full House

`eleccion_slot(Datos, Tablero, ia_det, full_house):-`

`puntaje(Datos, full_house, 25),`

`ocupada_categoria(full_house, Tablero, 0).`

Si no se logra ninguna de las combinaciones anteriores, la función comprueba si los dados forman un Full House (tres de un tipo y dos de otro) y si esta categoría está disponible en el tablero. De cumplirse ambos criterios, se elige la categoría full_house.

5. Mejor Categoría Superior

eleccion_slot(Dados, Tablero, ia_det, Categoria):-

mejor_categoria_superior(Dados, Tablero, Categoria).

Si las categorías Yahtzee, Large Straight, Small Straight y Full House no están disponibles o no son aplicables, la función elige la mejor categoría superior (aces, twos, threes, fours, fives, sixes) disponible en el tablero mediante la evaluación de los dados.

6. Four of a Kind

eleccion_slot(Dados, Tablero, ia_det, four_of_a_kind) :-

puntaje(Dados, four_of_a_kind, X),

X > 0,

ocupada_categoria(four_of_a_kind, Tablero, 0).

Si no se pueden aplicar las reglas anteriores, la función verifica si los dados forman un Four of a Kind (cuatro dados iguales) y si esta categoría está libre. Si es así, se elige la categoría four_of_a_kind.

7. Three of a Kind

eleccion_slot(Dados, Tablero, ia_det, three_of_a_kind) :-

puntaje(Dados, three_of_a_kind, X),

X > 0,

ocupada_categoria(three_of_a_kind, Tablero, 0).

Si no se pueden aplicar las reglas anteriores, la función verifica si los dados forman un Three of a Kind (tres dados iguales) y si esta categoría está libre. Si es así, se elige la categoría three_of_a_kind.

8. Chance

eleccion_slot(_, Tablero, ia_det, chance) :-

ocupada_categoria(chance, Tablero, 0).

Si ninguna de las categorías específicas está disponible o aplicable, la función verifica si la categoría chance está libre. Si es así, se elige esta categoría.

7. Categoría Aleatoria No Ocupada

eleccion_slot(_, Tablero, ia_det, X) :-

tomar_random_no_ocupada([chance, three_of_a_kind, four_of_a_kind, aces, twos, threes, fours, fives, sixes, full_house, small_straight, large_straight, yahtzee], Tablero, X).

Finalmente, si todas las categorías anteriores están ocupadas o no aplicables, la función elige aleatoriamente una categoría no ocupada en el tablero entre las posibles.

ia_prob:

Para la ia_prob se diseñó un código de problog para evaluar las probabilidades de obtener cierta categoría dada una configuración de dados con el fin de contemplar esta decisión a la hora de escoger una categoría, para lograr esto se definieron 5 predicados probabilísticos con valores entre 1 y 6 tal que cada uno tiene una probabilidad de 1/6 para simular el funcionamiento de cada dado.

La semántica que seguimos tiene la siguiente forma

categoria_with_n

tal que este predicado define las combinaciones donde una **categoria** dado que **n** dados coincidieron.

Por ejemplo, la probabilidad de obtener un yahtzee dado que tenemos 3 dados iguales es **yahtzee_with_3** :- dado4(X), dado5(X).

puntaje(+Dados, +Cat,-Puntos)

Tiene la misma funcionalidad que en ia_det.

puntaje_tablero(+Tablero, -Puntaje)

Tiene la misma funcionalidad que en ia_det.

ajustar_tablero(+Tablero,+Categoria,+Puntaje,-TableroSalida).

Tiene la misma funcionalidad que en ia_det.

cambio_dados(+Dados,+Tablero,+Estrategia,-Patron)

La IA probabilística (ia_prob) cambia los dados basándose en las siguientes reglas:

1. Categorías directas:
 - Si los dados forman un yahtzee (50 puntos) y la categoría no está ocupada, no se cambian los dados.
 - Similarmente para large_straight (40 puntos), small_straight (30 puntos) y full_house (25 puntos).
2. Sin repetidos:
 - Si los dados no tienen repetidos, se cambian todos los dados ([1,1,1,1,1]).
3. Máximas repeticiones:
 - Si hay repeticiones, se determina el valor más repetido (N) y se reemplazan todos los dados excepto aquellos con el valor N.
4. Probabilidades:
 - Se calculan las probabilidades de formar diferentes combinaciones (yahtzee, large_straight, small_straight, full_house) y se elige el mejor patrón de cambio de dados basándose en estas probabilidades.

eleccion_slot(+Datos,+Tablero,+Estrategia,-Categoria)

Tiene la misma funcionalidad que en ia_det.

Función prob

Como se mencionó anteriormente, la función prob calcula las probabilidades de concretar diferentes categorías basándose en la configuración de dados actuales, tal que,

1. Yahtzee:
 - Si la categoría yahtzee está ocupada, la probabilidad es 0.
 - Si hay 4, 3, 2, o 1 dados con el mismo valor, se calculan las probabilidades correspondientes (yahtzee_with_4, yahtzee_with_3, yahtzee_with_2, yahtzee_with_1).
2. Full House:
 - Similarmente, calcula las probabilidades de formar un full_house basado en las repeticiones de los dados (full_house_with_3, full_house_with_2).
3. Large Straight:
 - Calcula las probabilidades de formar una large_straight basándose en el tamaño de los huecos en la secuencia de dados (large_straight_h_2, large_straight_h_1).
4. Small Straight:
 - Calcula las probabilidades de formar una small_straight de manera similar a large_straight (small_straight_h_2, small_straight_h_1).

Tabla comparativa

| Nº de Partida | Seed | ia_det | ia_prob |
|---------------------|----------|------------|------------|
| 1 | 34826 | 209 | 209 |
| 2 | 77740 | 138 | 179 |
| 3 | 17392 | 203 | 183 |
| 4 | 1309 | 148 | 175 |
| 5 | 68599 | 194 | 161 |
| 6 | 81657 | 217 | 253 |
| 7 | 68501 | 149 | 163 |
| 8 | 66331 | 129 | 139 |
| 9 | 52248179 | 210 | 227 |
| 10 | 81657 | 217 | 253 |
| 11 | 79945 | 99 | 154 |
| 12 | 24975 | 122 | 132 |
| 13 | 21501 | 146 | 204 |
| 14 | 18933 | 195 | 230 |
| 15 | 35990 | 171 | 171 |
| 16 | 74921 | 193 | 178 |
| 17 | 52280 | 234 | 239 |
| 18 | 99702 | 166 | 162 |
| 19 | 50350 | 180 | 156 |
| 20 | 50961 | 128 | 230 |
| Media | | 172,4 | 189,9 |
| Desviación estándar | | 38,2311529 | 37,7915334 |