C311

INFORME DEL PROYECTO DE PROGRAMACION DECLARATIVA

Integrantes:

- Dianelys Cruz Mengana
- Jordan Pla González

UNIVERSIDAD DE LA HABANA | Facultad de Matemática y Computación

Para comenzar la simulación del proyecto se debe escribir en la terminal la instrucción `<simula(Cantidad_de_Participantes)>`

Cada color está representado por distintas cláusulas, de la siguiente forma:

```
%definicion de los azulejos
color(0,negro).
color(1,azul).
color(2,amarillo).
color(3,rojo).
color(4,celeste).
```

Componentes Principales:

- Fábricas y Centro
  Ambos están representados con los predicados `dynamic`: `fabrica/2`, `centro/2` respectivamente.
  En el caso de `<fabrica(i_Fabrica,Contenido)>` indica que la i-ésima fábrica está formada por la lista especificada en `Contenido`. Dicha lista contiene en el j-ésimo índice la cantidad de azulejos del color j-ésimo.
  En el caso de `<centro(Contenido)>` su definición es análoga a la de fábricas.

- Bolsa y Tapa
  Su representación es análoga al centro, igualmente con una lista `contenido`

- Tablero
  Está representado con los predicados `dynamic`: `puntuacion/2`, `area_de_preparacion/4`, `muro/3`, `suelo/1`.
  - `<puntuacion(i_Jugador,Score)>` especifica la puntuación del jugador i.
  - `<area_de_preparacion(i_Jugador, j_Fila, Cantidad, Azulejo)>` especifica el patrón del jugador i-ésimo en la fila j-ésima, la cantidad de azulejos puestos y por supuesto, el tipo de azulejo insertado.
  - `<muro(i_Jugador, j_Fila, Contenido)>` especifica para el muro del jugador i-ésimo en la fila j-ésima los azulejos insertados.
  - `<suelo(i_Jugador,Floor)>` especifica la cantidad de azulejos insertados en el suelo del jugador i-ésimo.

Sobre la simulación del juego el método principal antes mencionado, inicializa los predicados dinámicos con valores **default** y llama al predicado que ejecuta las distintas rondas siempre que no haya finalizado partida: `partida_acabada(1).`
`%1 simula el True”`

Para ejecutar todas las rondas usamos el predicado `ejecuta_ronda()`

```
%ejecutando rondas
ejecuta_ronda():-
    partida_acabada(1),
    cant_de_jugadores(J),
    calcular_puntos_adicionales(),
    determinar_ganadores(Ganadores),
    limpia_simulacion(),!.
ejecuta_ronda():-
    cant_de_ronda(R),
    RN is R + 1,
    retract(cant_de_ronda(_)),
    asserta(cant_de_ronda(RN)),
    preparar_ronda(),
    jugador_actual(Jugador),
    proxima_jugada(Jugador),
    ejecuta_ronda().
```

Mientras no se cumpla `partida_acabada(1)` cada vez que se intente unificar con `ejecuta_ronda()` el predicado que conduciría al éxito es el segundo, en donde se actualiza el número de rondas, dando paso al predicado `preparar_ronda()` en la cual se modifican solamente los predicados dinámicos necesarios para continuar a la siguiente ronda, estableciéndolos con valores por **default** en dependencia de su funcionalidad, se rellenan las fábricas, y se rellena la bolsa si es necesario.
NOTA: Para modificar dinámicamente los predicados, utilizamos `asserta/1`, `retract/1` y `retractall/1`.
Al unificar con el predicado `proxima_jugada(Jugador)` donde a través de llamados recursivos a dicho predicado, se ejecutan todos los jugadas de esa ronda por cada uno de los jugadores.
Luego realizamos el llamado al propio predicado, para repetir el mismo procedimiento y ejecutar la próxima ronda.

Una vez modificado `partida_acabada/1`, cualquier llamado al predicado `ejecuta_ronda()` unifica con la primera definición del predicado, según las reglas de Prolog, donde se calculan los puntos adicionales dependiendo de la cantidad de filas, columnas y colores completos, se determina el o los ganadores y limpiamos la simulación. Aquí hacemos uso del corte para que una vez haya triunfado este predicado, finalice la simulación.

En el predicado `juega(Jugador)` generamos las posibles jugadas para un jugador determinado según las reglas del juego. A continuación, explicaremos la **estrategia** empleada para la elección de las mismas.

La estrategia es greedy con determinadas consideraciones: a medida que se generan las jugadas se califican teniendo en cuenta si se completa una fila o una columna, así como la puntuación que se generaría en el muro atendiendo también a los azulejos que se dirigen al suelo.
Para simular la jugada óptima utilizamos el predicado dinámico `jugada_optima(Area, Azulejo, Patron_o_Suelo)`

- ✓ `Area` representa de donde obtiene el azulejo, que puede ser una fábrica o el centro.
- ✓ `Patron_o_Suelo` representa donde se coloca la ficha que puede ser en una fila del área de preparación o en el suelo.