Proyecto de Prolog.
Juego Hive.

Amalia Ibarra Rodríguez
Gabriela B. Martínez Giraldo

Universidad de La Habana,
La Habana, Cuba
Grupo C-412
{amalia.ibarra,gabriela.martinez}@estudiantes.matcom.uh.cu

1. Descripción del problema

El presente proyecto pretende ofrecer una impletención del juego Hive en el lenguaje de programación Prolog. Este es un juego de mesa muy curioso, en el cual cada jugador cuenta con 11 piezas que representan diferentes insectos a escoger entre: *abeja reina*(1), *hormiga*(3), *araña*(2), *escarabajo*(2), *bicho bola*(1), *mariquita*(1), *mosquito*(1) y *saltamontes*(3).

A continuación se presentan las ideas seguidas para modelar las situaciones requeridas por dicho juego.

2. Consideraciones sobre la disposición del tablero

En Hive no existe un tablero fijo, sino que se va expandiendo a medida que avanza el juego. Este está conformado por casillas hexagonales que pueden ser ocupadas por los insectos. A partir de que se coloca el primero, el tablero va creciendo a su alrededor. Se asume que la primera baldosa que aparece en el juego es la 0,0 y a partir de ella se definen sus adyacentes como:

(r-1, c)

(r, c-1)

(r+1, c-1)

(r+1, c)

(r, c+1)

(r-1, c+1)

si se consideran siguiendo el sentido de las manecillas del reloj y empezando por la casilla superior izquierda.

De esta forma, un tablero iría quedando como:

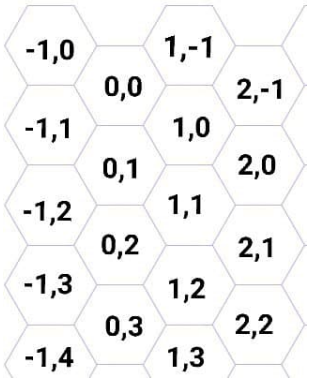

Figura 1. Coordenadas del tablero

3. Cómo correr el proyecto

En una consola dentro de la carpeta /src:

1. Ejecutar el comando swipl main.pl
2. Una vez en el entorno de swipl ejecutar start().

Este último inicializa los predicados que mantienen el estado de la partida. A partir de este punto comienza el juego y se pueden insertar y mover fichas a través de lo predicados:

1. insert_to(Bug).
   Aquí solo es necesario el nombre del insecto (que se especifica en Bug) pues se utiliza para insertar la primera ficha del juego que por defecto se ubica en (0,0).

2. insert_to(Bug, X, Y).
   Inserta el insecto (Bug) en la posición (X, Y).

3. move_to(X1, Y1, X2, Y2).
   Mueve la ficha ubicada en (X1, Y1) hacia la posición (X2, Y2).

4. move_to(X1, Y1, X2, Y2, X3, Y3).
   El bicho bola en la posición (X1, Y1) mueve la ficha ubicada en (X2, Y2) hacia la posición (X3, Y3).

5. move_to(X1, Y1, X2, Y2, X3, Y3).
   El mosquito ubicado en (X1, Y1) se mueve hacia (X3, Y3) usando el poder del insecto qu ese halla en (X2, Y2).

6. move_to(X1, Y1, X2, Y2, X3, Y3, X4, Y4).
   El mosquito en la posición (X1, Y1) utiliza el poder del bicho bola en (X2, Y2) para mover la ficha en (X3, Y3) hacia la posición (X4, Y4).

Dichos predicados actualizarán el estado del juego y evaluarán *true* en caso de ejecutar un movimiento válido o *false* en el caso contrario. Los insectos que se aceptan como entrada son: *queen, spider, ant, ladybug, pillbug, mosquito, grasshopper* y *beetle*.

4. Aspectos claves de la implementación

Para tener control del estado del juego fue necesario definir un número de predicados dinámicos que llevaran el conocimiento de la situación.

Así se define una ficha en una posición dada según el predicado *tile(Bug, Colour, X, Y, Level)*, donde *Bug* es el nombre del insecto, *Colour* su color, *X* y *Y* sus coordenadas en el tablero y *Level* su nivel. Este último valor se utiliza para representar la existencia de fichas apiladas (recordemos que el escarabajo es capaz de subir encima de otros insectos).

Se tiene constancia además de la cantidad de movimientos efectuados durante el juego a través de *move_count/1*, el cual determina qué turno corresponde a cada jugador, considerando que las blancas son las primeras en jugar.

Para limitar el número de insectos de cada tipo se definió el predicado *max_count/2*, el cual recibe el nombre de una especie específica y retorna la cantidad máxima que se pueden insertar. A su vez la cantidad de fichas de cada tipo colocadas en el tablero por cada jugador se registran a partir del predicado *insertion_count/2*.

4.1. Movimientos de cada insecto

En general ningún insecto puede ser insertado en una posición ocupada del tablero, ni en una donde colinde con fichas de color contrario. Tampoco puede añadirse a una casilla fuera de la colmena. Estas comprobaciones se llevan a través de los predicados *position_available/2*,*validate_insertion/4* que reciben los detalles necesarios del insecto a insertar.

Las reglas exigen insertar la *abeja reina* antes del 4to turno, y detallan que no es posible mover ninguna pieza antes de que esta esté en juego. Con este fin se hizo uso del predicado *queen_in_game/1* que no hace más que consultar en la BD de prolog la existencia de una reina de un color dado antes de mover cualquier insecto.

Como cada insecto tiene sus particularidades a la hora de moverse fue necesario implementar un predicado específico para cada uno, dado que sólo se

comparten pocas especificaciones como la de no tener otro insecto encima en ese momento y la de no romper la colmena. Estas se llevan con los predicados *pilled/2* y *one_hive_rule_fullfill/5*. Analicemos un poco más de cerca qué estrategia se siguió para verificar los movimientos válidos, definidos en su mayoría en el archivo *moves.pl*:

1. *Queen bee*: La abeja reina sólo da un paso en cada turno, por lo que para validar una candidata a próxima posición sólo se requiere definir las direcciones hacia las que se encuentran las celdas adyacentes y comprobar que la candidata sea una de estas y esté disponible. Como parte de los requerimientos generales se debe comprobar también que no exista un puente que bloquee este paso, lo cual nos facilita el predicado *not_blocked/5*.
2. *Beetle*: Su movimiento es similar al de la reina, sólo se omite la comprobación de que la celda esté disponible, dado que el poder especial de este insecto es precisamente saltar sobre otros.
3. *Pillbug*: Puede moverse un único paso hasta posiciones libres, completamente igual a la reina. En este se añade un predicado extra que permita al pillbug hacer uso de su poder, mover fichas adyacentes. Para ello se considera que no pueden existir puentes entre el primer insecto adyacente y el bicho bola ni entre este último y la posición destino.
4. *Ant*: La hormiga puede hacer recorridos más largos alrededor de la colmena, topando siempre con celdas ocupadas. La forma de comprobar la existencia de dicho camino entre un origen y un destino dado fue a través de un BFS o recorrido primero en anchura como se muestra a continuación, comprobando que no existan puentes que bloqueen su paso.

```
bfsft(Source, Destination):-
   bfsfta([Source], [Source], Destination).
bfsfta([[X,Y]|_R], _Seen, [X, Y]).

bfsfta([[X,Y]|R], Seen, [Xd, Yd]):-
   bagof([X2, Y2],
   (free_adj_tile((X,Y),(X2, Y2)),
   not_blocked(X, Y, X2, Y2, _Bridges),
   not(member([X2,Y2], Seen))), Adj),
   append(Seen, Adj, UpdSeen),
   append(R, Adj, UpdR), !,
   bfsfta(UpdR, UpdSeen, [Xd, Yd]);
   bfsfta(R, Seen, [Xd, Yd]).
```

5. *Spider*: Similar al caso anterior, su recorrido es bordeando la colmena, sólo que está limitao a una cantidad de 3 casillas en total, por lo que se utiliza una modificación del predicado anterior.
6. *Lady Bug*: De igual forma la mariquita requiere de un método para comprobar la existencia de un camino de tamaño 3 entre dos casillas dadas, sólo que

en este caso las dos primeras deben ser sobre baldosas ocupadas por otros insectos y la última una libre. Esta no requiere la comprobación de si existen puentes en el camino, dado que su ventaja es precisamente el entrar y salir de posiciones bloqueadas.

7. *Grasshopper*: El saltamontes puede saltar por un grupo de insectos que se encuentren dispuestos en línea recta en cualquier dirección, horizontal o vetical. Con *validate_grasshoper_move/4* se verifica que exista un camino que cumpla las condiciones anteriores, consultando la dirección y sentido del movimiento que se pretende efectuar y validando luego que exista en cada paso de ese camino un insecto en el tablero.
8. *Mosquito*: El mosquito es una pieza bastante versátil, pues puede adquirir el poder de cualquier otra ficha en juego que se encuentre adyacente a él. A su vez, cuando se haya logrado el comportamiento correcto del resto de los insectos este se hace bastante sencillo, dado que sólo requiere replicar el comportamiento de las mismas.