INTELIGENCIA ARTIFICIAL

Trabajo Práctico Nº 2: Cálculo Situacional

WATCH DRONE

Construcción de un agente inteligente que utiliza Cálculo Situacional.

Grupo II

José Ignacio Gómez

Pablo Nicolás Hechim

Sebastián Federico Paciuk

Índice

Gómez, José Ignacio – Hechim, Pablo Nicolás – Paciuk, Sebastián Federico

Trabajo práctico Nº2 – "WATCH DRONE"

Grupo 11

José Ignacio Gómez – gomez.ignacio31@gmail.com
Pablo Nicolás Hechim – nicolashechim@gmail.com
Sebastián Federico Paciuk – sebapaciuk@gmail.com

Resumen. El presente trabajo de la cátedra Inteligencia Artificial, consiste en diseñar e implementar un agente inteligente basado en conocimiento. Este tendrá un objetivo definido. Para alcanzarlo contará con una serie de acciones de las cuales el agente se basará en un razonamiento determinado para cierta situación. Las situaciones varían, y en base a éstas, el agente debe inferir cual es la mejor acción a ejecutar.

"WATCH DRONE" es un programa de computadora diseñado para implementar un agente "drone" que utiliza cálculo situacional para encontrar personas que cometieron un hecho ilícito. El objetivo del "drone" es obtener la ubicación exacta de dichas personas antes que logren escapar de su alcance.

La presente es la segunda y última entrega del trabajo práctico.

1. Introducción

En este trabajo se considerará una ciudad donde un *DroneAgent* percibirá señales que serán disparadas sobre personas por un sistema de alertas en un ambiente nano-radio. Éstas alertas serán accionadas como consecuencia de un hecho ilícito y pondrán al *DroneAgent* en búsqueda, donde éste deberá distinguir entre víctimas y victimarios para así encontrar al victimario en cuestión. El cambio más grande respecto a la etapa anterior del trabajo práctico está en el agente, debido a que esta vez será un agente basado en conocimiento, donde razonará sobre los resultados de sus propias acciones hasta poder llegar a su objetivo. Además, el escenario será más pequeño y el agente sólo se desplazará sobre el nivel de altitud más bajo.

En la próxima sección: "Solución", se detallará la solución implementada, y las aclaraciones pertinentes. A continuación, en "Resultados", mostraremos algunos de los resultados obtenidos al interactuar con el *DroneAgent*. Luego, en "Conclusiones", indicaremos qué conclusión se puede derivar luego de hacer funcionar el *DroneAgent* con cálculo situacional.

2. Solución

2.1 Consideraciones previas

Procederemos a documentar la solución conceptual del problema.

En primer lugar, presentaremos al agente, sin olvidar que:

> "Un agente es un sistema que está situado en algún ambiente, y que es capaz de realizar **acciones** autónomas en ese ambiente, para cumplir sus **objetivos** de diseño. Para que un agente sea inteligente debe ser *proactivo*, *reactivo* y tener *habilidad social.*"

Nuestro drone, entonces, será *DroneAgent* y percibirá el entorno a través de su *cámara* y *GPS*, y decidirá qué acciones pertinentes debe llevar a cabo para distinguir y encontrar un victimario. Notar que la percepción de *antena* que teníamos en la primera etapa se ha eliminado, dado que el drone solo se mueve en el nivel bajo y damos por sentado que ya ha percibido de dónde provienen las señales.

El ambiente de nuestro agente, será lo que el *DroneAgent* percibe. Incorporará conocimiento a partir de sus percepciones, y ejecutará una acción racional a partir de su base de conocimiento que modificará el ambiente. Se necesita que realice un proceso de cálculo situacional cada vez que perciba señales del ambiente. El comportamiento se ilustra en la Figura 1.

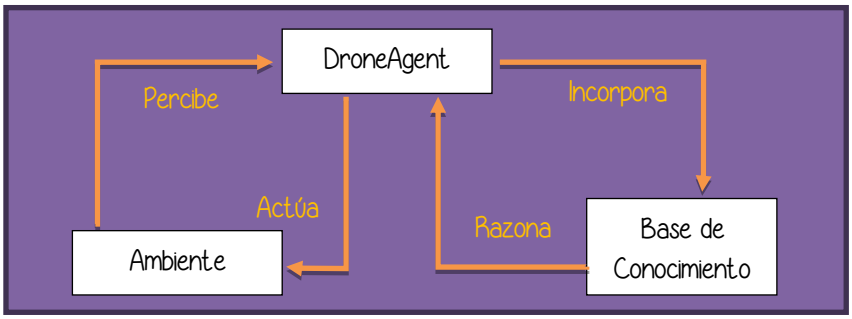

Figura 1 – Interacción Agente-Ambiente

En la Figura 2 se muestra un diagrama IDEM-IA que modela el estado del agente, las operaciones que este puede realizar, el estado del ambiente, las percepciones del agente y la función objetivo definida.

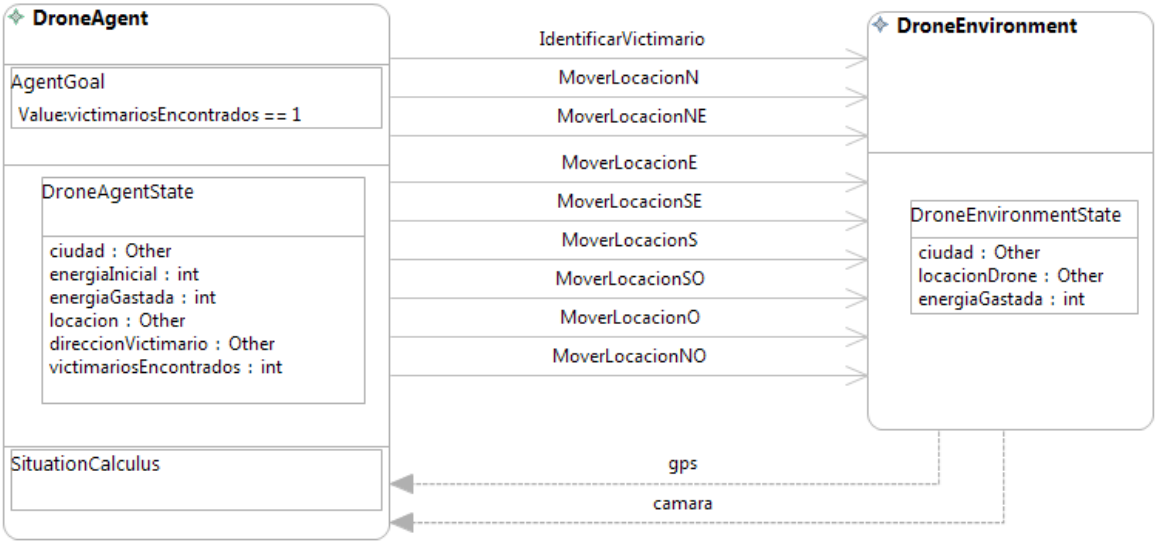

Figura 2 – Diagrama IDEM-IA

2.2 Definición de representación de estados

Estado del agente:

El estado del agente está representado por medio de seis parámetros o variables como se muestra en la Figura 3.

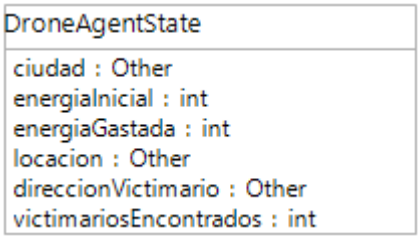

Figura 3 – Representación del estado del agente

Donde:

- ✓ **ciudad**: Estructura de datos que representa el mapa donde *DroneAgent* puede moverse, y sólo posee los datos que éste conoce.
- ✓ **energíaInicial**: Representa las unidades de energía inicial que dispone *DroneAgent*.
- ✓ **energíaGastada**: Representa el consumo total acumulado de energía que realiza *DroneAgent* al ejecutar acciones.
- ✓ **locación**: Representa la posición actual de *DroneAgent* dentro del mapa (**ciudad)**.

- ✓ **direcciónVictimario**: Indica si hay un victimario en cada una de las posiciones cardinales respecto a la posición actual de *DroneAgent*.
- ✓ **victimariosEncontrados**: Indica la cantidad de victimarios encontrados por *DroneAgent*.

Estado inicial del agente:

- ✓ **ciudad**: Contiene cargadas la estructura de cuadrante – subcuadrante – esquina. No contempla información sobre ubicaciones de víctimas y victimarios.
- ✓ **energíaInicial**: Se inicializa en 1000 unidades de energía.
- ✓ **energíaGastada**: Se inicializa en 0 unidades de energía.
- ✓ **locación**: Se inicializa en una esquina seleccionada por medio de la interfaz que muestra el mapa **(ciudad)**.
- ✓ **direcciónVictimario**: Se inicializan todas las posiciones cardinales en false.
- ✓ **victimariosEncontrados**: Se inicializa en 0.

Estado final del agente:

- ✓ **ciudad**: Contiene cargadas la estructura de cuadrante – subcuadrante – esquina. Contempla la ubicación de víctimas y victimarios percibidos por *DroneAgent* durante la búsqueda del objetivo.
- ✓ **energíaInicial**: 1000 unidades de energía.
- ✓ **energíaGastada**: Indica la cantidad de energía consumida por *DroneAgent* hasta llegar al objetivo.
- ✓ **locación**: Indica la última ubicación de *DroneAgent*. Es decir, donde encontró al victimario.
- ✓ **direcciónVictimario**: Contiene la dirección en la que se encuentra el victimario respecto a la posición de *DroneAgent*. Ésta corresponde a la última percepción que *DroneAgent* obtuvo con su cámara cuando alcanzó el objetivo.
- ✓ **victimariosEncontrados**: Indica la cantidad de victimarios encontrados, en este caso, será un único victimario.

Estado del ambiente

El estado del ambiente está representado por medio de tres parámetros o variables como se muestra en la Figura 4.

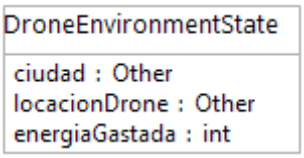

Figura 4 – Representación del estado del ambiente

Donde:

- ✓ **ciudad**: Estructura de datos que representa el mapa donde *DroneAgent* puede moverse. Contempla las intensidades, cantidades y ubicaciones de víctimas y victimarios.
- ✓ **locaciónDrone**: Indica la ubicación actual de *DroneAgent*.
- ✓ **energíaGastada**: Indica la energía gastada hasta el momento por *DroneAgent*.

2.3 Definición de percepciones

- ✓ **GPS**: Permite a *DroneAgent* saber cuál es su ubicación actual.
- ✓ **Cámara**: Permite a *DroneAgent* identificar a un victimario. Le muestra en línea recta todas las calles desde la posición actual hasta el límite del cuadrante en el cual se encuentra, pudiendo así decidir en qué dirección moverse para encontrar un victimario.

2.4 Definición de reglas diagnósticas

Las reglas diagnosticas permiten a través de las percepciones determinar propiedades del ambiente. Tenemos tres tipos de reglas diagnósticas, una que determina si hay un victimario en la esquina actual (Cámara), la otra si hay un victimario en alguna dirección cardinal (Cámara) y por último la posición actual de *DroneAgent* (GPS).

```
1. hayVictimario(Esq,S):-
   perception(_,_,_,_,_,_,_,_,1,Xpos,Ypos),esquina(Esq,Xpos,Ypos),actualSituation
   (S).

2. direccionVictimario(N,NE,E,SE,Su,SO,O,NO,PosAct,S):-
   perception(N,NE,E,SE,Su,SO,O,NO,PosAct,_,_),actualSituation(S).

3. posicion(Xpos,Ypos,S):-
   perception(_,_,_,_,_,_,_,_,_,Xpos,Ypos),actualSituation(S).
```


Gómez, José Ignacio – Hechim, Pablo Nicolás – Paciuk, Sebastián Federico

2.5 Valoración de acciones

Se establecen cuatro valoraciones:

- Excelente: Se considera así a la acción que permite que *DroneAgent* identifique al victimario. Esto se da cuando se encuentra en una esquina con victimario.
- Muy buena: Se considera así a la acción que permite que *DroneAgent* se desplace en una dirección en la que hay un victimario.
- Buena: Se considera así a la acción que permite que *DroneAgent* se desplace hacia una esquina con personas pero sin victimario en esa dirección.
- Regular: Se considera así a la acción que permite que *DroneAgent* se desplace hacia una esquina vacía y sin victimario en esa dirección.

Reglas:

1. `excelent(identificarVictimario,S):- posible(identificarVictimario,S).`
2. `very_good(moverLocacionN,S):- direccionVictimario(1,_,_,_,_,_,_,_,_,S),posible(moverLocacionN,S).`
3. `very_good(moverLocacionNE,S):- direccionVictimario(_,1,_,_,_,_,_,_,_,S),posible(moverLocacionNE,S).`
4. `very_good(moverLocacionE,S):- direccionVictimario(_,_,1,_,_,_,_,_,_,S),posible(moverLocacionE,S).`
5. `very_good(moverLocacionSE,S):- direccionVictimario(_,_,_,1,_,_,_,_,_,S),posible(moverLocacionSE,S).`
6. `very_good(moverLocacionS,S):- direccionVictimario(_,_,_,_,1,_,_,_,_,S),posible(moverLocacionS,S).`
7. `very_good(moverLocacionSO,S):- direccionVictimario(_,_,_,_,_,1,_,_,_,S),posible(moverLocacionSO,S).`
8. `very_good(moverLocacionO,S):- direccionVictimario(_,_,_,_,_,_,1,_,_,S),posible(moverLocacionO,S).`
9. `very_good(moverLocacionNO,S):- direccionVictimario(_,_,_,_,_,_,_,1,_,S),posible(moverLocacionNO,S).`
10. `good(moverLocacionN,S):- direccionVictimario(0,_,_,_,_,_,_,_,_,S),posible(moverLocacionN,S),posicion(Xpos,Ypos,S),esquina(Esq,Xpos,Ypos),adyacente(Esq,Esq1,norte),potencia(Esq1,N),N > 0.`
11. `good(moverLocacionNE,S):- direccionVictimario(_,0,_,_,_,_,_,_,_,S),posible(moverLocacionNE,S),posicion(Xpos,Ypos,S),esquina(Esq,Xpos,Ypos),adyacente(Esq,Esq1,norEste),potencia(Esq1,N),N > 0.`

```
12. good(moverLocacionE,S):-
    direccionVictimario(_,_,0,_,_,_,_,_,_,S),posible(moverLocacionE,S),posicion(Xp
    os,Ypos,S),esquina(Esq,Xpos,Ypos),adyacente(Esq,Esq1,este),potencia(Esq1,N),N
    > 0.

13. good(moverLocacionSE,S):-
    direccionVictimario(_,_,_,0,_,_,_,_,_,S),posible(moverLocacionSE,S),posicion(X
    pos,Ypos,S),esquina(Esq,Xpos,Ypos),adyacente(Esq,Esq1,surEste),potencia(Esq1,N
    ),N > 0.

14. good(moverLocacionS,S):-
    direccionVictimario(_,_,_,_,0,_,_,_,_,S),posible(moverLocacionS,S),posicion(Xp
    os,Ypos,S),esquina(Esq,Xpos,Ypos),adyacente(Esq,Esq1,sur),potencia(Esq1,N),N >
    0.

15. good(moverLocacionSO,S):-
    direccionVictimario(_,_,_,_,_,0,_,_,_,S),posible(moverLocacionSO,S),posicion(X
    pos,Ypos,S),esquina(Esq,Xpos,Ypos),adyacente(Esq,Esq1,surOeste),potencia(Esq1,
    N),N > 0.

16. good(moverLocacionO,S):-
    direccionVictimario(_,_,_,_,_,_,0,_,_,S),posible(moverLocacionO,S),posicion(Xp
    os,Ypos,S),esquina(Esq,Xpos,Ypos),adyacente(Esq,Esq1,oeste),potencia(Esq1,N),N
    > 0.

17. good(moverLocacionNO,S):-
    direccionVictimario(_,_,_,_,_,_,_,_,0,_,S),posible(moverLocacionNO,S),posicion(X
    pos,Ypos,S),esquina(Esq,Xpos,Ypos),adyacente(Esq,Esq1,norOeste),potencia(Esq1,
    N),N > 0.

18. regular(moverLocacionN,S):- posible(moverLocacionN,S).

19. regular(moverLocacionNE,S):- posible(moverLocacionNE,S).

20. regular(moverLocacionE,S):- posible(moverLocacionE,S).

21. regular(moverLocacionSE,S):- posible(moverLocacionSE,S).

22. regular(moverLocacionS,S):- posible(moverLocacionS,S).

23. regular(moverLocacionSO,S):- posible(moverLocacionSO,S).

24. regular(moverLocacionO,S):- posible(moverLocacionO,S).

25. regular(moverLocacionNO,S):- posible(moverLocacionNO,S).


26. bestAction(noAction,S):- goalReached(S),!.

27. bestAction(X,S):- excelent(X,S),!.

28. bestAction(X,S):- very_good(X,S),!.

29. bestAction(X,S):- good(X,S),!.

30. bestAction(X,S):- regular(X,S),!.
```

```
31. goalReached(S):- cantidadVictimarios(N,S), N > 0.
```

2.6 Axiomas de posibilidad

Los axiomas de posibilidad definen cuándo se puede ejecutar una acción. Se tienen dos tipos de axiomas:

- identificarVictimario: si *DroneAgent* se encuentra en una esquina donde hay un victimario, es posible llevar a cabo esta acción.
- moverLocacion: si *DroneAgent* quiere moverse en una dirección en la que hay una esquina adyacente y cuenta con la energía necesaria que insume este movimiento, es posible llevar a cabo esta acción.

Axiomas:

```
1. posible(identificarVictimario,S):-                    posicion(Xpos,Ypos,S),
   esquina(Esq,Xpos,Ypos),hayVictimario(Esq,S).

2. posible(moverLocacionN,S):-
   posicion(Xpos,Ypos,S),esquina(Esq1,Xpos,Ypos),adyacente(Esq1,Esq2,norte),energ
   ia(EDisp,S),costo(Esq2,C),EDisp >= C,not(visitada(Esq2)).

3. posible(moverLocacionNE,S):-
   posicion(Xpos,Ypos,S),esquina(Esq1,Xpos,Ypos),adyacente(Esq1,Esq2,norEste),ene
   rgia(EDisp,S),costo(Esq2,C),EDisp >= C,not(visitada(Esq2)).

4. posible(moverLocacionE,S):-
   posicion(Xpos,Ypos,S),esquina(Esq1,Xpos,Ypos),adyacente(Esq1,Esq2,este),energi
   a(EDisp,S),costo(Esq2,C),EDisp >= C,not(visitada(Esq2)).

5. posible(moverLocacionSE,S):-
   posicion(Xpos,Ypos,S),esquina(Esq1,Xpos,Ypos),adyacente(Esq1,Esq2,surEste),ene
   rgia(EDisp,S),costo(Esq2,C),EDisp >= C,not(visitada(Esq2)).

6. posible(moverLocacionS,S):-
   posicion(Xpos,Ypos,S),esquina(Esq1,Xpos,Ypos),adyacente(Esq1,Esq2,sur),energia
   (EDisp,S),costo(Esq2,C),EDisp >= C,not(visitada(Esq2)).

7. posible(moverLocacionSO,S):-
   posicion(Xpos,Ypos,S),esquina(Esq1,Xpos,Ypos),adyacente(Esq1,Esq2,surOeste),en
   ergia(EDisp,S),costo(Esq2,C),EDisp >= C,not(visitada(Esq2)).

8. posible(moverLocacionO,S):-
   posicion(Xpos,Ypos,S),esquina(Esq1,Xpos,Ypos),adyacente(Esq1,Esq2,oeste),energ
   ia(EDisp,S),costo(Esq2,C),EDisp >= C,not(visitada(Esq2)).

9. posible(moverLocacionNO,S):-
   posicion(Xpos,Ypos,S),esquina(Esq1,Xpos,Ypos),adyacente(Esq1,Esq2,norOeste),en
   ergia(EDisp,S),costo(Esq2,C),EDisp >= C,not(visitada(Esq2)).
```

2.7 Axiomas de estado sucesor

Los axiomas de estado sucesor permiten agregar nuevos conocimientos a partir de las acciones que ejecute el agente. Se tienen tres tipos de axiomas:

```
% Mantiene de situación en situación si hayVictimario en una esquina.
```

1.
```
est(S1) :-
        S1 > 0, S is S1-1,
        hayVictimario(Esq,S),
        asserta(hayVictimario(Esq,S1)).
```

```
% Aumenta de situación en situación la cantidadVictimarios si se identificó un victimario en la última acción.
```

2.
```
est(S1) :-
        S1 > 0, S is S1-1,
        cantidadVictimarios(N,S),
        not(executedAction(identificarVictimario,S)),
        asserta(cantidadVictimarios(N,S1)).
```

```
% Mantiene de situación en situación la cantidadVictimarios si no se identificó un victimario en la última acción.
```

3.
```
est(S1) :-
        S1 > 0, S is S1-1,
        cantidadVictimarios(N,S),
        N1 is N + 1,
        executedAction(identificarVictimario,S),
        asserta(cantidadVictimarios(N1,S1)).
```

```
% Si se mueve, cambia de posición.
```

4.
```
est(S1) :-
        S1 > 0, S is S1-1,
        posicion(Xpos,Ypos,S),
        esquina(Esq,Xpos,Ypos),
        adyacente(Esq,Esq1,norte),
        esquina(Esq1,Xpos1,Ypos1),
        executedAction(moverLocacionN,S),
        asserta(posicion(Xpos1,Ypos1,S1)).
```

5.
```
est(S1) :-
        S1 > 0, S is S1-1,
        posicion(Xpos,Ypos,S),
        esquina(Esq,Xpos,Ypos),
        adyacente(Esq,Esq1,norEste),
        esquina(Esq1,Xpos1,Ypos1),
        executedAction(moverLocacionNE,S),
        asserta(posicion(Xpos1,Ypos1,S1)).
```

6.
```
est(S1) :-
        S1 > 0, S is S1-1,
        posicion(Xpos,Ypos,S),
        esquina(Esq,Xpos,Ypos),
```

```
        adyacente(Esq,Esq1,este),
        esquina(Esq1,Xpos1,Ypos1),
        executedAction(moverLocacionE,S),
        asserta(posicion(Xpos1,Ypos1,S1)).

7.  est(S1) :-
        S1 > 0, S is S1-1,
        posicion(Xpos,Ypos,S),
        esquina(Esq,Xpos,Ypos),
        adyacente(Esq,Esq1,surEste),
        esquina(Esq1,Xpos1,Ypos1),
        executedAction(moverLocacionSE,S),
        asserta(posicion(Xpos1,Ypos1,S1)).

8.  est(S1) :-
        S1 > 0, S is S1-1,
        posicion(Xpos,Ypos,S),
        esquina(Esq,Xpos,Ypos),
        adyacente(Esq,Esq1,sur),
        esquina(Esq1,Xpos1,Ypos1),
        executedAction(moverLocacionS,S),
        asserta(posicion(Xpos1,Ypos1,S1)).

9.  est(S1) :-
        S1 > 0, S is S1-1,
        posicion(Xpos,Ypos,S),
        esquina(Esq,Xpos,Ypos),
        adyacente(Esq,Esq1,surOeste),
        esquina(Esq1,Xpos1,Ypos1),
        executedAction(moverLocacionSO,S),
        asserta(posicion(Xpos1,Ypos1,S1)).

10. est(S1) :-
        S1 > 0, S is S1-1,
        posicion(Xpos,Ypos,S),
        esquina(Esq,Xpos,Ypos),
        adyacente(Esq,Esq1,oeste),
        esquina(Esq1,Xpos1,Ypos1),
        executedAction(moverLocacionO,S),
        asserta(posicion(Xpos1,Ypos1,S1)).

11. est(S1) :-
        S1 > 0, S is S1-1,
        posicion(Xpos,Ypos,S),
        esquina(Esq,Xpos,Ypos),
        adyacente(Esq,Esq1,norOeste),
        esquina(Esq1,Xpos1,Ypos1),
        executedAction(moverLocacionNO,S),
        asserta(posicion(Xpos1,Ypos1,S1)).

% Si no se mueve, mantiene la posición.

12. est(S1) :-
        S1 > 0, S is S1-1,
        posicion(Xpos,Ypos,S),
```

```
        executedAction(identificarVictimario,S),
        asserta(posicion(Xpos,Ypos,S1)).
```

% Disminuye de situación en situación la energía disponible cuando la última acción ejecutada fue de movimiento.

```
13. est(S1) :-
        S1 > 0, S is S1-1,
        energia(E,S),
        posicion(Xpos,Ypos,S1),
        esquina(Esq,Xpos,Ypos),
        costo(Esq,Costo),
        not(executedAction(identificarVictimario,S)),
        E1 is E - Costo,
        asserta(energia(E1,S1)).
```

% Mantiene de situación en situación la energía disponible cuando la última acción ejecutada fue identificarVictimario.

```
14. est(S1) :-
        S1 > 0, S is S1-1,
        energia(E,S),
        executedAction(identificarVictimario,S),
        asserta(energia(E,S1)).
```

% Marca como visitada la esquina a la cual se movió el agente.

```
15. est(S1) :-
        S1 > 0, S is S1-1,
        posicion(Xpos,Ypos,S),
        esquina(Esq,Xpos,Ypos),
        posicion(Xpos1,Ypos1,S1),
        esquina(Esq1,Xpos1,Ypos1),
        Esq \= Esq1,
        asserta(visitada(Esq1)).
```

2.8 Prueba de meta

Se considera que se alcanzó el objetivo si se encuentra un victimario:

```
goalReached(S):- cantidadVictimarios(N,S), N > 0.
```

3. Resultados

En esta sección se presenta un escenario de prueba en el cual se puede apreciar el accionar de *DroneAgent* utilizando cálculo situacional.

El escenario usado corresponde al de la figura 5; en la misma pueden distinguirse:

- ✓ La posición inicial de WatchDrone: se representa con un círculo de mayor tamaño respecto al resto de las esquinas.
- ✓ Esquinas sin señal: son representadas con puntos de color azul.
- ✓ Esquinas con señal: se representan con puntos de color verde.
- ✓ Esquinas con señal y victimarios: se representan con puntos de color rojo.
- ✓ Los límites del subcuadrante en los cuales se moverá WatchDrone (el cuadrado más interno).

Figura 5 – Escenario de prueba

A continuación se presenta las acciones ejecutadas por WatchDrone en su proceso de encontrar al victimario:

1- MoverLocacionO (a A4M4B1).
2- MoverLocacionS (a A4M4B7).
3- MoverLocacionS (a A4M4B6).
4- MoverLocacionE (a A4M4B5).

5- MoverLocacionE (a A4M4B4).
6- IdentificarVictimario.

Siguiendo las acciones ejecutadas se puede apreciar cómo *DroneAgent* responde a la evaluación de acciones propuesta.

Inicialmente *puede* ir hacia el Este, Sur y Oeste. Según el orden de las reglas de estado sucesor, debería ir al Este en primera instancia. Esto le permitiría llegar más rápido al objetivo, ya que si primero se mueve al Este y luego al Sur, estará en la misma esquina que el victimario.

Sin embargo, elige ir en un principio hacia el Oeste (A4M4B1) ya que en esa esquina hay *señal* y esta acción está considerada como una *buena* acción, y mejor que moverse hacia una esquina sin personas que está considerada una acción *regular*. Aquí podemos observar como las reglas de su conocimiento le indican que "es mejor" ir hacia una esquina con señal que sin señal (principalmente por el costo de movimiento, y porque se supone que estará más cerca de hallar el victimario).

En la situación siguiente elige ir al Sur (A4M4B7) por la misma razón, es decir, en esa dirección hay una esquina con personas; la otra opción es ir al Este pero en dicha esquina no hay personas

Luego va hacia el Sur de nuevo (A4M4B6); ¿por qué ejecuta esta acción si ir a una esquina sin señal es "peor" que ir a una que sí tiene, como la esquina al Norte de su posición actual? Pues la respuesta es sencilla; el Drone tiene una regla que le indica que no puede volver a pasar por una esquina si ya ha sido *visitada*. Por este motivo, su único camino es hacia el Sur.

Una vez en la esquina A4M4B7, debe elegir entre las próximas 2 acciones que *puede* realizar: ir al Este e ir al Sur. Uno supondría que como la esquina hacia el Este no posee señal y la esquina hacia el Sur sí, WatchDrone elegiría ir hacia el Sur. Pero lo que WatchDrone decide en base a su conocimiento es ir hacia el Este. ¿Por qué? Una vez más, las reglas le indican que esa es la mejor acción. Como WatchDrone a percibido a través de la cámara que hacia el Este encontrará un victimario en alguna de las esquinas de esa dirección, decide aventurarse en ese camino. Es decir, moverse en la dirección en la que se detectó que hay un victimario es una acción *muy buena*, y mejor que ir al Sur hacia una esquina con señal que es una acción *buena*. Así es como primero se mueve al Este a la esquina A4M4B5 y luego, por el mismo motivo, al Este hacia la esquina A4M4B4.

Una vez en A4M4B4, percibe que en su posición actual hay un victimario. Luego, decide *identificar cuál de ellas es el victimario*, dado que esta acción es considerada como una acción *excelente*. Esta acción se ejecutará siempre que WatchDrone sepa que está en una esquina en que hay un victimario y aún no haya alcanzado su objetivo.

4. Comparación entre búsqueda y cálculo situacional.

Con ambas técnicas *DroneAgent* logra alcanzar el objetivo para los escenarios propuestos. En cuanto a las implementaciones de ambos, vemos como la estrategia de búsqueda forma un árbol en cada situación y cada nodo se fija cual es la mejor opción para ir a cualquier nodo adyacente. En cambio, en cálculo situacional, el agente percibe, decide y acciona según lo que crea más conveniente, almacenando la información de la situación actual en una base de conocimiento.

En otras palabras, con búsqueda con cada paso que el Drone da, debe recalcular de cero cuál es el mejor camino a la solución; con cálculo situacional, en cambio, el Drone puede ir almacenando cierta información que le resulta útil para llegar más rápido al objetivo (es decir, puede ir basándose en conocimiento adquirido).

Otra diferencia importante que encontramos, es que en búsqueda el Drone *explora*; ejecuta todos los operadores que puede y decide cual es la secuencia de acciones que lo lleva a su objetivo, y ejecuta la primera de ellas; en el caso del cálculo situacional, el Drone elige la próxima acción a ejecutar basado en la situación actual, los conocimientos previos, sus *reglas* de operación y un ranking de las posibles acciones a ejecutar.

Teniendo en cuenta esto, y el tiempo de implementación de la solución en cada uno de los casos, decimos que nos resultó más conveniente usar Cálculo Situacional.

5. Conclusiones

La realización del presente trabajo nos permitió llevar la teoría al ámbito práctico para la construcción de un agente inteligente. Resultando muy fructífero desde el punto de vista que asentó bases de conocimiento, demandó análisis intensos y mucha interrelación de los contenidos vistos en clases. Desde la práctica logramos asimilar de forma más directa cómo funciona la estrategia de cálculo situacional en un caso real.

Consideramos que el modelo propuesto para resolver problema fue bueno y determinante para el desarrollo del trabajo práctico. Nos resultó fácil de usar, intuitivo y de mucha ayuda. Es una buena manera de modelar un sistema, ya que, busca explotar los rasgos característicos de un agente y su relación con el ambiente.

Positivamente sentimos haber alcanzado el objetivo planteado: *"comprender cómo se relaciona el agente inteligente con el mundo en el que desenvuelve y cómo utiliza las técnicas aprendidas para la toma de decisiones sobre las acciones que puede emprender."*

6. Bibliografía

1. S. Russell, P. Norvig; **Inteligencia Artificial. Un enfoque moderno.** Prentice Hall (1996).
2. Ejemplos de Agentes Inteligentes implementados, ofrecidos por la cátedra Inteligencia Artificial, FRSF, UTN.