

A continuación tienes el enunciado del ejercicio entregable sobre agentes reflex, el cual se realizará mediante un Notebook. Para aportar tu solución para los apartados (i) y (ii) del ejercicio I, usa la celda dispuesta para ello al final del *ipynb*.


I) Aspirador como Agente reflex

I Versión simplificada, dos habitaciones

Diseña un **agente reflex simple** que represente un aspirador automático en un entorno de dos habitaciones con las siguientes especificaciones:

- El agente tiene dos sensores: estado de limpieza (limpio/sucio) y localización (habitación izquierda/derecha).
- El agente puede seleccionar una de entre tres acciones en cada iteración: aspirar, moverse hacia la derecha o moverse hacia la izquierda.


Se pide:

- Enumera los estados posibles del sistema.
- Realiza un diagrama completo de las transiciones entre estados.
-  Codifica el entorno en python, y programa un agente reflex que resuelva el problema. El agente, comenzando en cualquier habitación, debe dejar todas las habitaciones limpias.
- El programa que usa el agente para seleccionar la acción a realizar debe de consistir **exclusivamente** en un conjunto de sentencias `if`, `elif`, `else`, cada una de las cuales implementa una regla de **condición-acción**:

```
1 def selectAction(percept):
2     # selects action based on current percept and condition-action
   rules
3     if "current percept meets condition1":
4         return "some action"
5     elif "current percept meets condition2":
6         return "some other action"
7     ...
```

II Versión unidimensional extendida

Extiende el modelo de dos habitaciones a una casa con n habitaciones en línea.

- Añade un sensor que detecte la presencia de una pared cuando el agente está en una de las habitaciones de los extremos (es decir, las acciones de moverse ahora están condicionadas por los sensores de "pared").
-  Codifica el entorno en python, y programa un agente reflex que resuelva el problema. El agente puede estar situado inicialmente en cualquier habitación, y debe dejar todas las habitaciones limpias.
- Además de los perceptos que recibe del entorno, el agente mantiene una **pequeña memoria**, que puede contener, a lo sumo, las acciones realizadas por el mismo en los **dos** pasos precedentes, y los perceptos recibidos en los **dos** pasos precedentes. La memoria no debe contener nada más.
- El programa que usa el agente para seleccionar la acción a realizar debe de consistir **exclusivamente** en un conjunto de sentencias `if`, `elif`, `else`, cada una de las cuales implementa una regla de **condición-acción**. Las condiciones pueden mencionar los perceptos actuales o la información presente en la memoria (perceptos y acciones precedentes):


```

1 def selectAction(percept):
2     # selects action based on current percept and condition-action
    rules
3     if "current percept meets condition1" and / or "percepts/actions in
        memory meet condition2":
4         return "some action"
5     elif "current percept meets condition3" and / or "percepts/actions
        in memory meet condition4":
6         return "some other action"
7     ...
8

```

III Versión bidimensional

Extiende el modelo del problema anterior una casa con $n \times m$ habitaciones formando un rectángulo.

- i. En este caso, además del sensor de estado de limpieza (limpio/sucio), añade un sensor de pared que indique si el agente tiene una pared **arriba**, **abajo**, a su **derecha** o a su **izquierda**. En las esquinas, el sensor indica la presencia de las dos paredes correspondientes.
- ii.  Codifica el entorno en python, y programa un agente reflex que resuelva el problema. El agente puede estar situado inicialmente en cualquier habitación, excepto aquellas que están junto a una pared, y debe dejar todas las habitaciones limpias.
- iii. Además de los perceptos que recibe del entorno, el agente mantiene una **pequeña memoria**, que puede contener, a lo sumo, las acciones realizadas por el mismo en los **dos** pasos precedentes, y los perceptos recibidos en los **dos** pasos precedentes. La memoria no debe contener nada más.
- iv. El programa que usa el agente para seleccionar la acción a realizar debe de consistir **exclusivamente** en un conjunto de sentencias **if**, **elif**, **else**, cada una de las cuales implementa una regla de **condición-acción**. Las condiciones pueden mencionar los perceptos actuales o la información presente en la memoria (perceptos y acciones precedentes):

```

1 def selectAction(percept):
2     # selects action based on current percept and condition-action
    rules
3     if "current percept meets condition1" and / or "percepts/actions in
        memory meet condition2":
4         return "some action"
5     elif "current percept meets condition3" and / or "percepts/actions
        in memory meet condition4":
6         return "some other action"
7     ...
8

```

Plantilla Agente reflex

Para realizar esta práctica debes usar las plantillas de los agentes correspondientes en el notebook adjunto en la entrega, respetando los formatos ahí presentes. Este notebook es accesible también es accesible a través de [este enlace](#). Como verás, en algunas celdas de ejecución se encuentran los *prints* de una solución posible. Estos deben servirte como referencia a la hora de resolver el ejercicio. Los perceptos posibles para los agentes vienen dados por el diccionario `percept` dentro de la función `getPerceptFromEnvironment`, y las acciones que pueden guardarse en memoria vienen indicadas en la función `setEnvironment`. Debes rellenar aquellas partes del código indicadas con:

```

""" YOUR CODE GOES HERE """

```