Tecnológico de Monterrey

Evidencia 1 Actividad Integradora

Modelación de sistemas multiagentes con gráficas computacionales

(Gpo 102)

Edwin Iñiguez Moncada A01637064

Santiago López Campos A01643411

Jesús Enrique Bañales López A01642425

Moises Adrián Cortés Ramos A01642492

Ernesto Puga Araujo A00572845

27/08/2024

Especificaciones de Propiedades de agentes

- RobotAgent:
    - Atributos:
        - ‘agentType’: Tipo de agente (0 para robot)
        - ‘carrying’: Referencia a la caja que el robot esta llevando (inicialmente ‘none’)
        - ‘moves’: Contador de movimientos realizados por el robot
        - ‘direction’: Dirección en la que el robot se mueve, (por ejemplo norte, sur, este y oeste)
    - Métodos:
        - ‘see(e)’: Percepción del entorno, detectando cajas cercanas
        - ‘next()’: Ejecución de acciones basadas en reglas
        - ‘move_N()’, ‘move_S()’, ‘move_E()’, ‘move_W()’, ‘move_random()’: Métodos para mover el robot en diferentes direcciones
        - ‘pickup()’: Función de limpieza
        - ‘turn()’: Cambiar la dirección de movimiento

- BoxAgent:
    - Atributos:
        - ‘agentType’: Tipo de agente (1 para box)
        - ‘first_step’: Indicador para saber si es el primer paso del agente
        - ‘pos’: Posición actual de la caja
    - Métodos
        - ‘setup()’: Inicializa la posición del agente
        - ‘step()’: Lógica de paso del agente

Propiedades del Entorno

- RobotModel:
    - Atributos:
        - ‘robots’: Lista de agentes tipo Robot
        - ‘boxes’: Lista de agentes tipo Box
        - ‘grid’: Grid que contiene los agentes
    - Métodos:
        - ‘setup()’: Inicializa el modelo creando agentes y agregándolos al grid.
        - ‘step()’: Lógica de paso del modelo, que ejecuta los pasos de todos los agentes

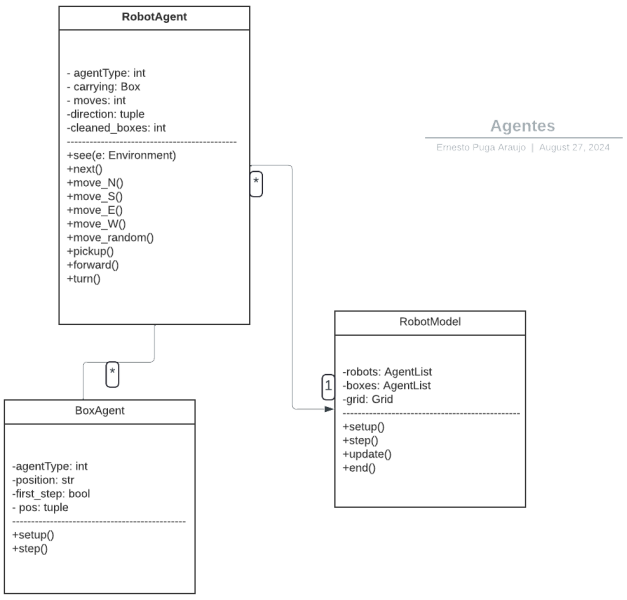
Métricas de Éxito

- RobotAgent:
    - Métrica de éxito:
        - Número de movimientos: Contar el número total de movimientos realizados por el robot. Menos movimientos pueden indicar mayor eficacia.
        - Eficiencia de Limpieza: Se puede medir como el porcentaje de suciedad eliminada respecto a la cantidad total de suciedad en el entorno. Se puede calcular así:

$$Eficiencia\ de\ limpieza\ = \frac{Total\ de\ suciedad\ eliminada}{Total\ de\ suciedad\ inicial} x100$$

- BoxAgent:
    - Métrica de éxito:
        - Cantidad de cajas recogidas: Se puede medir como la cantidad de cajas recogidas por el robot durante la simulación.

Diagramas de clases

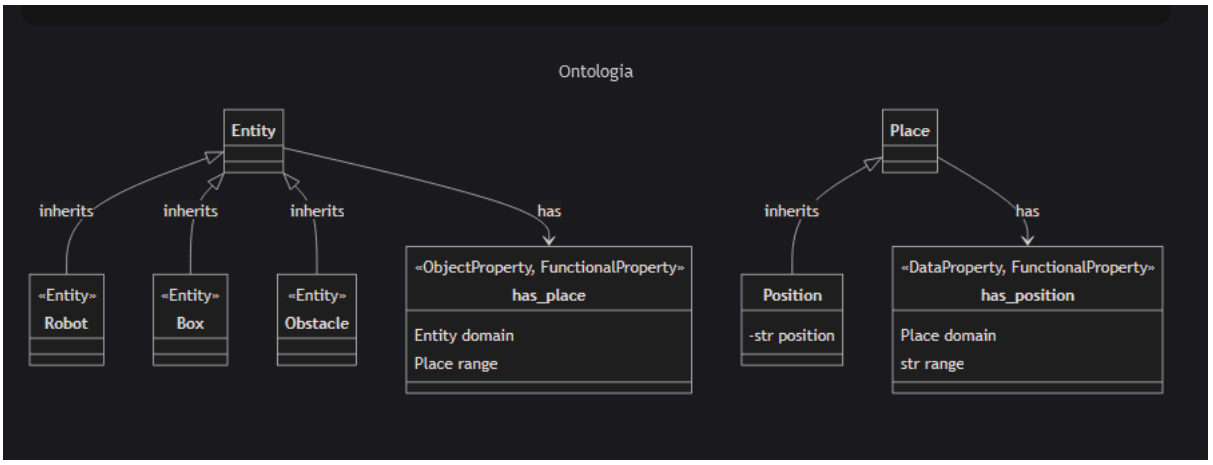


Relaciones entre clases:

- ‘RobotAgent’ y ‘BoxAgent’:
    - Relación: interacción

    - Descripción: Los agentes de tipo ‘RobotAgent’ interactuán con los agentes de tipo ‘BoxAgent’. Ya que un ‘RobotAgent’ puede recoger o apilar ‘BoxAgent’ en la simulación. Esto se representa mediante métodos en ‘RobotAgent’, como ‘pickup()’.
- ‘RobotModel’ y ‘RobotAgent’:
    - Relación: Composición
    - Descripción: ‘RobotModel’ contiene y gestiona una lista de ‘RobotAgent’. El modelo crea instancias de agentes robots y los agrega a la cuadrícula, lo que índica que ‘RobotModel’ es responsable de la creación y gestión del ciclo de vida de los ‘RobotAgent’.
- ‘RobotModel’ y ‘BoxAgent’:
    - Relación: Composición
    - Descripción: También contiene y gestiona una lista de ‘BoxAgent’. Esto significa que ‘RobotModel’ controla la creación y la gestión de los agentes de tipo ‘BoxAgent’ dentro de la simulación.

Diagrama de ontología



- Clases principales:
    - Entity:
        - Es la clase base de la ontología. Representa cualquier objeto que pueda existir en el entorno de la simulación .
    - Robot:
        - Hereda de Entity. Representa los robots en la simulación, que son agentes capaces de moverse y realizar tareas como recoger las cajas y apilarlas.
    - Box:
        - También hereda de Entity. Representa las cajas que los robots deben recoger y apilar en la simulación.
    - Obstacle:

        - De igual manera hereda de Entity. Representa cualquier objeto en el entorno que podría interferir con los movimientos de los robots.
    - Place:
        - Es una clase independiente que representa una ubicación en el entorno . Un Place podría ser cualquier lugar donde una Entity podría ser situada.
    - Position:
        - Hereda de Place. Es una clase que especifíca la posición dentro del entorno. Está asociada con un atributo de tipo str que podría representar coordenadas.

Propiedades

- has_place:
    - Es una propiedad funcional que asocia una Entity con un Place. Indica que cada entidad debe estar en un lugar específico del entorno.
- has_position:
    - Es una propiedad funcional que asocia un Place con un Position. Indica la ubicación exacta de un lugar en el entorno.

Relaciones

- Herencia:
    - Las clases Robot, Box y Obstacle heredan de la clase Entity, lo que significa que comparten características comunes definidas en Entity.
- Asociación:
    - La propiedad has_place crea una relación de asociación entre Entity y Place.
    - La propiedad has_position asocia un Place con un Position, especificando su ubicación en el entorno.

Conclusión

En este proyecto, hemos desarrollado un sistema de simulación de agentes utilizando AgentPy como framework, para modelar el comportamiento de robots en un entorno grid, complementado con la ontología que estructura las entidades involucradas, como los robots, las cajas y los obstáculos. El sistema también incluye un conjunto de reglas y acciones que permiten a los agentes tomar decisiones en función de su entorno.

Para mejorar la eficiencia de los agentes en este sistema, se podrían considerar las siguientes alternativas:

- Implementación de Algoritmos de Búsqueda Heurística:

    - Utilizar algoritmos como Dijkstra para optimizar las rutas que los robots siguen en el entorno. Estos algoritmos permitirían a los robots planificar movimientos más eficientes, evitando los obstáculos y reduciendo el tiempo necesario para completar tareas.
- Aprendizaje por Refuerzo:
    - Integrar técnicas de aprendizaje por refuerzo para que los robots puedan aprender a mejorar su comportamiento a lo largo del tiempo, adaptándose mejor a situaciones dinámicas y complejas.
- Distribución Inteligente de Tareas:
    - Desarrollar un sistema de coordinación y asigncación de tareas entre los robots, asegurando que las tareas se distribuyan equitativamente y que los recursos se utilicen de manera óptima.
- Modelos Predictivos:
    - Implementar modelos predictivos que anticipen posibles cambios en el entorno, permitiendo que los agentes ajusten su comportamiento de manera proactiva en lugar de reactiva.

Cada una de estas alternativas nos pueden ofrecer la posibilidad de mejorar la eficiencia de los agentes en distintas dimensiones, ya sea en términos de velocidad, precisión, adaptabilidad o colaboración. La elección de la solución más adecuada dependera de las características especificas del entorno y las necesidades del sistema en cuestión.