Modelado Actividad M3

Modelación de Sistemas Multiagentes con Gráficas Computacionales – TC2008B.4

Esteban de la Maza

A01720548

Introducción

El propósito de esta tarea es describir el modelo que se utilizara para una intersección vehicular donde existe comunicación entre múltiples agentes de alto nivel. Se definirán los agentes que estarán involucrados, así como su descripción y la interacción que tendrán con el resto de los agentes. El entorno donde se encontrarán será de tipo **cooperativo** pues la meta de los agentes es de llegar a su destino y minimizar los accidentes vehiculares. Sera **parcialmente observable**, dado que los vehículos no conocerán toda la información del resto de los carros en todo instante. Sera **determinista** dado que la posición de los vehículos depende de la posición previa. Sera **secuencial, dinámico y continuo**. Además, será **Multiagente** debido a que habrá muchos agentes tomando parte en el modelo.

La intersección que se modelará en este caso será donde se cruzan 2 calles perpendiculares, teniendo un semáforo en cada dirección y vehículos buscando dirigirse en cualquier dirección excepto de la que vienen. Es decir, no habrán vueltas en U. El desarrollo tendrá una estructura Orientada a Objetos, donde cada tipo de agente será representado por una clase heredada de las clases de la librería MESA, ya sea Agente o Modelo.

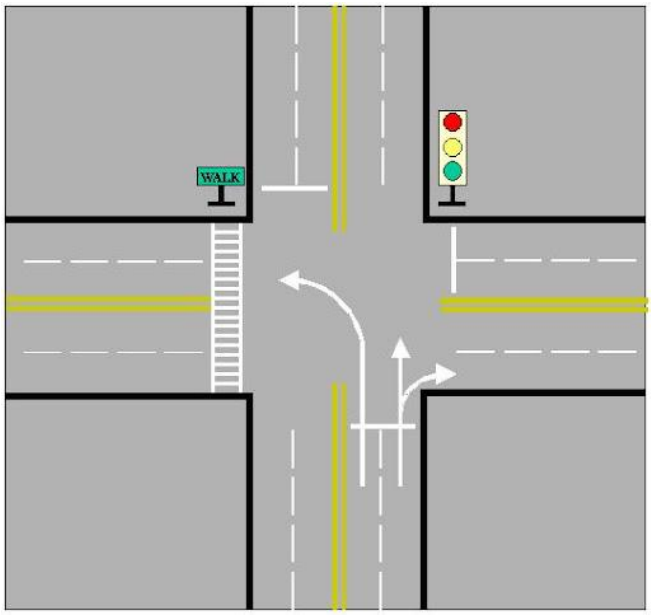

Ilustración 1Intersección que se Modelará

Requisitos del Modelo

Mientras no haya un vehículo cercano, el semáforo estará en luz amarilla.

Cuando un vehículo se acerque a la intersección, enviará un mensaje con el tiempo estimado de arribo.

El semáforo dará luz verde al semáforo más cercano y establecerá un programa de luces a partir de ese punto para el resto de los vehículos.

Agentes Involucrados

El primer agente que modelare y el más importante son los **vehículos**. Estos serán quienes tendrán que cruzar la intersección, con el objetivo de llegar a su destino y minimizar la cantidad de accidentes que ocurren en esta. Mas del 50% de los accidentes automovilísticos ocurren en intersecciones (NSW, 2021) debido a que existen muchas reglas que todos los conductores deben de conocer y respetar. Solo se necesita un conductor que no las conozca y puede causar un choque y frenar la vialidad por completo. Estos conductores deberán de ser capaces de comunicarse entre sí y con los semáforos para determinar cuándo es su turno. Adicionalmente, deberán de conocer su destino para así poder establecer una ruta que los llevara a ese lugar. Además, deberán de ser capaces de recibir mensajes por parte de los semáforos para entender cuando es su turno de avanzar y cuando deberán de esperar. Finalmente, deberán de tener presente a sus "vecinos" para así saber si puede avanzar hacia donde se quiere dirigir o si se encuentra un vehículo en esa posición del tablero.

Los vehículos tendrán la posibilidad de conducir por 3 direcciones al llegar a la intersección. Podrán seguir en su camino y avanzar directamente hacia adelante, podrán dar vuelta hacia la izquierda o derecha. Los semáforos estarán programados para que solo exista un verde y 3 rojos. Los autos que se encuentren en la dirección del verde podrán tomar cualquiera de las 3 opciones.

Otro agente crucial para el funcionamiento del cruce serán los **semáforos**. Estos serán inteligentes de tal forma que optimicen el flujo vehicular y minimicen el tiempo de espera de los carros. Los semáforos se encontrarán en color amarillo hasta que no se acerque un vehículo y entonces tenga que analizar la situación. Posteriormente, tendrá que detectar los vehículos y enviarles un mensaje con el tiempo estimado de arribo. Finalmente, uno de los 4 semáforos se pondrá en Verde y el resto deberán de permanecer en Rojo para evitar una colisión. Mientras se encuentre en Verde uno y en Rojo el resto, deberá de continuar de analizar el flujo para así realizar su siguiente decisión y determinar cuál dirección será la siguiente en avanzar. Para analizar cuantos carros se encuentran en la fila de dicho semáforo, se va a necesitar la ayuda de un agente tipo sensor**.** Estos van a pertenecer a cada semáforo y se encontraran a lo largo de la fila que corresponde a cada uno. Así, los semáforos tendrán información sobre cuantos carros se encuentran en su cola y compararlo con el de los demás.

El estado de los semáforos se guardará en un arreglo global por fuera de la clase donde todos los agentes pueden leer los estados actuales de todos los semáforos. De esta forma los vehículos sabrán cuando pueden cruzar la calle en la que se encuentran.

Los vehículos tienen que ser capaces de detectar en donde se encuentran. Para esto, necesitaremos modelar un agente de tipo **piso**. Existen 2 tipos de piso el primero siendo calle. Estos se encontrarán en aquellas partes del tablero donde se encuentra pavimentado y es permitido que los autos transiten por ahí. El resto del tablero será un agente de tipo banqueta para así establecer en donde los vehículos no pueden circular y establecer los límites del tablero.

Finalmente, se va a necesitar un agente tipo **controlador** quien estará a cargo de manejar la información de los 4 semáforos, tanto el tiempo que llevan desde el ultimo cambio, así como la fila actual que se encuentra en cada punto de la intersección. Este agente llamara a las funciones de cada semáforo para realizar los cambios de estado para controlar el flujo vehicular.

Atributos de cada Agente

Vehículos:

- int posX: Posición actual en X
- int posY: Posición actual en Y
- int destinationX: Posición de su destino en X
- int destinationY: Posición de su destino en Y
- Agent[] neighbors: Arreglo de agentes de sus vecinos
- int lightState: Entero con valor que representa el estado del semáforo que le corresponde

Semáforo

- int state: Entero que representa su estado actual
    - Se utilizarán para reconocer los colores los valores a continuación:
        - 0: Rojo
        - 1: Amarillo
        - 2: Verde
- Time timeSpent: Valor de tipo tiempo que representa el tiempo que lleva en su estado actual
- int lineSize: Este entero guardara el valor que regresa la función de los sensores para saber cuantos carros hay en su fila actualmente.

Piso

- int posX: Posición actual en X
- int posY: Posición actual en Y
- bool isLimit: Determina si está ubicado justo antes del cruce para saber si debe de alertar a los vehículos
- bool isStreet: Para saber de que tipo es el Piso:
    - True: Calle
    - False: Banqueta

Sensor Semáforo:

- int posX: Posición actual en X
- int posY: Posición actual en Y
- bool hasCar:
    - True: Si hay un carro
    - False: Esta vacío el lugar

Controlador

- No tendrán atributos

Métodos de Cada Agente

Los agentes tipo **vehículo** tendrán los siguientes métodos:

- Agent[] getNeighbors(): Vehículos en su alrededor y Tipo de Piso
    - A través de una función donde analice a sus vecinos y los guarde en un arreglo. Se utilizará la función que proporciona la librería MESA. Si encuentra un vehículo enfrente de su posición, deberá de permanecer parado y no avanzará.
    - Adicionalmente, estarán verificando por cuales posiciones en el tablero puede avanzar dependiendo si son de tipo calle o banqueta.
- int getLightState(): Función para obtener el Estado del Semáforo que dirige el Sentido en el que se Encuentran
    - Al acceder a la variable global donde se guardan los estados.
- void move(): Función para avanzar y acercarse a su destino final.
    - Dado que cada agente cuenta con un par de coordenadas de su destino final, esta función determinara hacía que dirección se debe de mover para acercarse a su destino. Esto se puede realizar a través de prueba y error con sus vecinos para determinar cuál posición lo acerca más a su destino. Aquí es donde tendrá que verificar con precisión si es banqueta o calle y la posición a la que se quiere mover y si existen carros en dichas posiciones para determinar si es un movimiento valido.
- stop(): Función para ser detenido
    - Esta función será llamada por parte de los agentes tipo piso que se encuentren en el borde de la intersección para poder detener a los automóviles y forzarlos a que esperen un Verde de los semáforos.

Los agentes tipo **semáforos** tendrán los siguientes métodos:

- int getLine(): Para Obtener la Cantidad de Carros en la fila que maneja
    - Esta tarea ser realizara a través de comunicación con los sensores que se encuentran a su disposición, para así determinar cuál semáforo tiene la fila más larga y entonces cambiar su estado actual. Regresaran un valor booleano los sensores si es que se encuentra un vehículo en su posición del tablero y esta función sumara la cantidad de veces que la respuesta sea un True.
- void changeState(int color): Función para actualizar el estado del semáforo
    - Le pasaran como parámetro un entero que represente un color de los 3 posibles.
    - Estos cambios se verán reflejados en el arreglo previamente mencionado que se encuentra accesible en todo el programa para guardar los estados actuales de los 4 semáforos.
    - Una vez que se genere un cambio guardará un *timestamp* en una variable global con el tiempo donde empezó el ultimo cambio de estado ocurrido. Una vez que se realice un cambio, se llamara esta función y se actualizara la variable.
- int[] getLightsArray(): Función para obtener el estado actual del resto de los semáforos
    - Acceder a la variable global que guarda el arreglo
- time getTimeSpent(): Esta función servirá para obtener el tiempo trascurrido del estado actual
    - Dentro de la función, se utilizará una variable para calcular el tiempo trascurrido desde el último cambio de estado del semáforo y el tiempo actual.

Los agentes tipo **piso** tendrán los siguientes métodos:

- void alertCar(): Si está en el borde del cruce, significa que tendrá que alertar a los automóviles y deberán de pedir el estado actual de su semáforo. Utilizaran una llamada a la función de stop() para lograrlo. Una vez que el vehículo obtenga un mensaje de que el semáforo se encuentra en verde, podrá avanzar.

Los agentes tipo **controlador** tendrán los siguientes métodos:

- void analize():
    - Esta función será de alta importancia pues será quien compilará la información que almacena cada semáforo para poder realizar la toma de decisiones. Aquí es donde se harán las comparaciones de la cola que tiene cada semáforo, así como el tiempo que lleve en algún estado. Una vez recopilada esta información, se harán llamadas a las funciones de cambio de estado de cada semáforo para controlar el flujo.
    - Además, esta será quien estimará el tiempo de arribo de cada semáforo que no este en verde para poder entregar este valor a cada semáforo.
    - Esta función será llamada dentro de la función de step() de la clase.

Los agentes tipo **Sensor Semáforo** tendrán los siguientes métodos:

- bool isVacant():
    - Esta función va a determinar si tiene un vehículo que se encuentra en su posición actualmente. Sera llamada por parte de los semáforos y ellos manejaran la información.

Diagrama de Clases

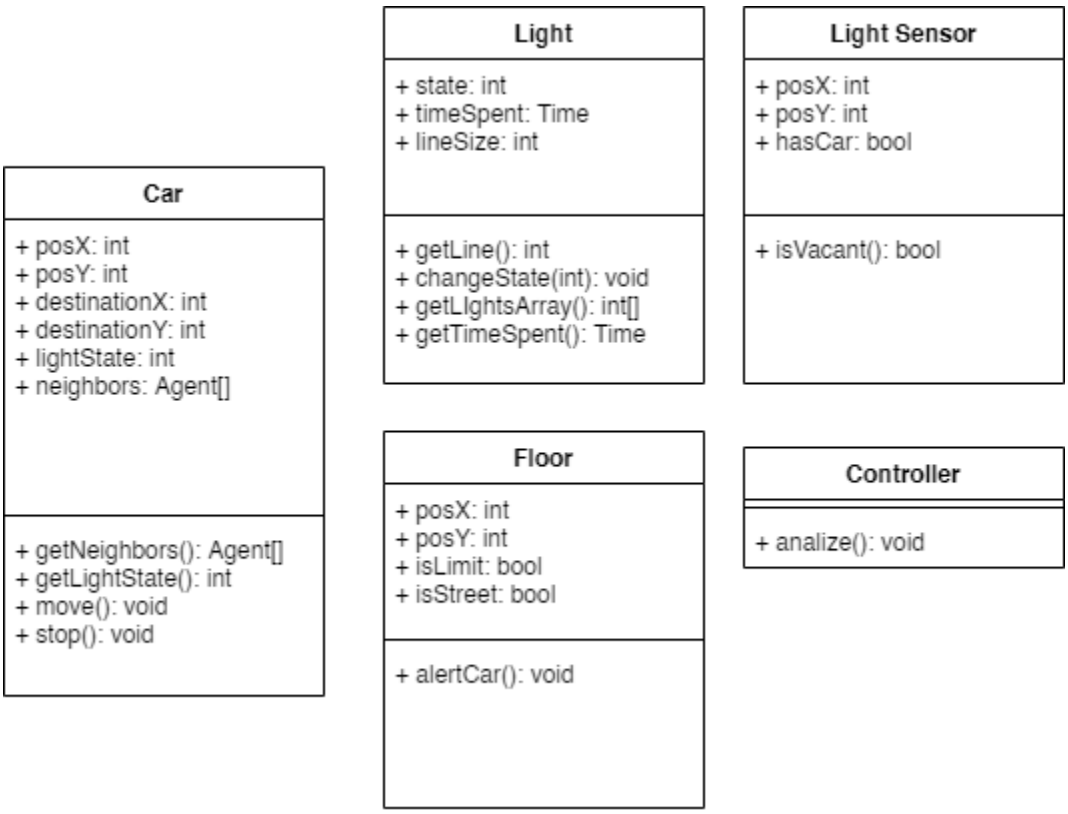

Ilustración 2 Diagrama de Clases

Referencias

Bauer, Bernhard, Jorg P. Muller y James Odell: Agent UML: A Formalism for Specifying Multiagent Software Systems. En Ciancarini, Paolo y Michael J. Wooldridge (editores): Agent-Oriented Software Engineering, páginas 91–103, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg, ISBN 978-3-540-44564-7.

Mesa: Agent-based modeling in Python 3+ — Mesa .1 documentation. (2021). Retrieved 23 August 2021, from https://mesa.readthedocs.io/en/stable/

NSW. (2021). Intersections. Safety and Rules. https://roads-waterways.transport.nsw.gov.au/roads/safety-rules/stopping-turning/intersections.html.