Breve ensayo sobre STRIPS PLANNER

Por Angel Garc**í**a B**á**ez

La inteligencia artificial (IA) ha buscado desde sus inicios desarrollar sistemas capaces de realizar tareas complejas que requieren decisiones secuenciales. En este contexto, uno de los principales desafíos ha sido crear sistemas capaces de planificar: generar secuencias de acciones que transformen un estado inicial en un estado objetivo deseado. El enfoque STRIPS (Stanford Research Institute Problem Solver), desarrollado en 1971 por Richard Fikes y Nils Nilsson, supuso un hito en este campo al proporcionar un método formal y eficiente para la planificación automática. STRIPS no solo ayudó a resolver problemas específicos de la robótica en su momento, sino que sentó las bases para una amplia variedad de aplicaciones en la planificación automatizada, desde la robótica hasta la toma de decisiones complejas en el ámbito de la logística y la simulación.

### Origen y estructura del sistema STRIPS

El sistema STRIPS fue diseñado en el Instituto de Investigación de Stanford (SRI) como parte de un proyecto pionero en la robótica, Shakey the Robot. Este robot tenía la capacidad de razonar sobre su entorno y planificar secuencias de acciones para alcanzar metas en un espacio físico. La necesidad de dotar a Shakey de una capacidad de planificación llevó a Fikes y Nilsson a desarrollar un lenguaje que permitiera representar el mundo y las acciones de manera formal y manejable.

El enfoque STRIPS se basa en tres componentes fundamentales: el estado inicial, que describe el conjunto de condiciones actuales del mundo; el estado objetivo, que define las condiciones deseadas al final de una secuencia de acciones; y los operadores o acciones, que se componen de precondiciones (aquellas que deben cumplirse para que la acción sea ejecutada) y efectos (los cambios que dicha acción provocará en el estado del mundo). Las acciones son representadas a través de predicados lógicos, y el algoritmo de planificación se encarga de encontrar la secuencia de acciones que transforme el estado inicial en el estado objetivo.

Esta representación estructurada y lógica permitió que el sistema STRIPS se usara no solo para controlar el comportamiento de robots como Shakey, sino que también se convirtió en el modelo base para una nueva generación de planificadores automáticos. Su formalismo hizo posible que las máquinas razonaran sobre secuencias de eventos de manera eficiente, lo cual representaba un avance significativo frente a enfoques anteriores que intentaban resolver problemas similares sin un marco lógico claro.

### Importancia de STRIPS en la IA

La aparición de STRIPS fue clave en el desarrollo de la planificación automática, ya que introdujo un modelo formalmente definido para la representación de problemas de planificación en IA. Este modelo no solo simplificó la formulación de problemas complejos, sino que también proporcionó un marco generalizable que podía adaptarse a diferentes dominios. STRIPS impulsó la investigación en planificación automatizada, motivando el desarrollo de algoritmos de búsqueda y optimización que permitieran explorar eficientemente el espacio de soluciones posibles.

Uno de los mayores logros de STRIPS fue la separación clara entre el estado del mundo y las acciones disponibles. Esta distinción permitió a los planificadores razonar sobre las acciones de manera más eficiente, ya que el estado del mundo se actualizaba dinámicamente a medida que se ejecutaban las acciones, lo que facilitaba la exploración de secuencias de acciones en entornos complejos. La simplicidad de la estructura de STRIPS lo hizo especialmente atractivo en aplicaciones de robótica, donde las decisiones debían tomarse en tiempo real con base en el entorno dinámico.

### Aplicaciones y evolución de STRIPS

Aunque STRIPS fue diseñado originalmente para la robótica, su estructura lógica y formal encontró aplicaciones en una amplia gama de campos. En la robótica moderna, STRIPS y sus derivados siguen utilizándose para que los robots planifiquen tareas como ensamblaje de piezas, navegación en entornos desconocidos y manipulación de objetos. En los videojuegos, los sistemas de planificación automática basados en STRIPS permiten que los personajes no jugables (NPCs) tomen decisiones lógicas basadas en su entorno, mejorando así la experiencia de juego.

Además, STRIPS ha sido empleado en la optimización logística para la planificación de rutas de entrega y la secuenciación de tareas en cadenas de suministro. Los planificadores automáticos basados en STRIPS también han sido fundamentales en la toma de decisiones en misiones espaciales y en la planificación militar, donde es necesario generar secuencias de acciones que maximicen la eficiencia y minimicen los riesgos.

A lo largo del tiempo, STRIPS ha evolucionado para adaptarse a problemas más complejos y dinámicos. Lenguajes de planificación como PDDL (Planning Domain Definition Language), que se utilizan hoy en día en competiciones internacionales de planificación automática, extienden las ideas de STRIPS, permitiendo la inclusión de tiempos, costos, y recursos, ampliando así la capacidad de modelado y planificación en escenarios más realistas.

### Conclusión

STRIPS marcó un antes y un después en la historia de la inteligencia artificial al proporcionar una solución formal y eficiente para el problema de la planificación automática. Su legado es evidente en una amplia gama de aplicaciones, desde la robótica hasta la logística y los videojuegos, donde la planificación secuencial es esencial para el éxito de los sistemas inteligentes. A través de STRIPS y sus sucesores, los sistemas de IA han logrado alcanzar niveles impresionantes de autonomía y eficiencia, lo que ha impulsado el desarrollo de soluciones aún más complejas y sofisticadas en el campo de la planificación automática. STRIPS sigue siendo un pilar fundamental en la IA moderna, demostrando la importancia de una representación clara y lógica en la resolución de problemas secuenciales.

**Ejercicios Capítulo 2**

**1. Explore las diferencias entre funciones de agente y programas de agente.**

**- ¿Puede haber más de un programa de agente que implemente una función de agente dada? Dé un ejemplo o muestre por qué no es posible.**

R: Sí, puede haber más de un programa de agente que implemente una función de agente dada. Esto se debe a que una función de agente describe el comportamiento en términos de "qué acción debe tomar el agente en respuesta a una percepción

Función de agente: Si la temperatura percibida es inferior a 20°C, enciende la calefacción; si es mayor de 25°C, enciende el aire acondicionado; en cualquier otro caso, no hagas nada.

**- ¿Existen funciones de agente que no puedan ser implementadas por ningún programa de**

**agente?**

Una función de agente computable es aquella que puede ser implementada por un programa de agente utilizando un algoritmo finito, es decir, el programa de agente puede procesar los perceptos y devolver una acción en un tiempo finito.

Sin embargo, existen funciones de agente no computables, que son aquellas que no pueden ser calculadas por ningún programa de agente porque implican problemas que están más allá de las capacidades de cualquier sistema computacional, incluso teóricamente.

**- Dada una arquitectura fija de máquina, ¿cada programa de agente implementa**

**exactamente una función de agente?**

La respuesta es sí, dado una arquitectura fija de máquina, cada programa de agente implementa exactamente una función de agente.

Dado que el programa de agente se ejecuta en una arquitectura fija y sigue reglas o algoritmos definidos de manera precisa, mapea cada posible conjunto de perceptos a exactamente una acción. Este mapeo constituye una única función de agente.

**- Dada una arquitectura con n bits de almacenamiento, ¿cuántos programas de agente**

**diferentes son posibles?**

Se pueden formar 2 a la n programas posibles basados en la representación de todas las combinaciones de los bits.

**- Supongamos que mantenemos el programa de agente fijo, pero aceleramos la máquina**

**por un factor de dos. ¿Cambia eso la función del agente?**

No, acelerar la máquina por un factor de dos no cambia la función del agente. La función del agente sigue siendo la misma, aunque el programa de agente se ejecute más rápidamente.

2. Por cada una de las siguientes afirmaciones, diga si es verdadera o falsa y apoye su respuesta

con ejemplos o contraejemplos donde sea apropiado.

**- Un agente que solo percibe información parcial sobre el estado no puede ser**

**perfectamente racional.**

**R:** La afirmación es falsa. Un agente que percibe información parcial sobre el estado sí puede ser perfectamente racional, siempre y cuando actúe de manera que maximice su rendimiento esperado dado el conocimiento limitado que tiene sobre el entorno.

Contraejemplo de un Agente No Racional con Información Parcial: Supongamos un agente que tiene información parcial y toma decisiones aleatorias o sin fundamento. Aunque su información es limitada, no está tomando decisiones óptimas según lo que sabe. Este agente no sería racional. La clave es que la racionalidad depende de cómo se usan las percepciones disponibles, no de cuántas percepciones tiene.

**- Existen entornos de tareas en los que ningún agente puramente reflejo puede comportarse de manera racional.**

**R:** La afirmación es **verdadera**. Existen entornos de tareas en los que ningún agente puramente reflejo puede comportarse de manera racional, debido a las limitaciones inherentes de los agentes reflejos simples.

En entornos muy simples donde las percepciones actuales son suficientes para tomar decisiones óptimas, un agente puramente reflejo puede ser racional. Por ejemplo, un termostato básico que enciende o apaga la calefacción según la temperatura actual puede ser perfectamente racional en su entorno, ya que solo necesita la percepción actual de temperatura para funcionar.

**- Existe un entorno de tareas en el que cada agente es racional.**

La afirmación es **falsa**. No existe un entorno de tareas en el que todos los agentes sean racionales, porque la racionalidad depende de cómo un agente toma decisiones en función de sus objetivos y del conocimiento que tiene del entorno. En la práctica, no todos los agentes tomarán decisiones óptimas en cualquier entorno debido a sus limitaciones, diseño o información incompleta.

**- La entrada a un programa de agente es la misma que la entrada a la función del agente.**

R: La afirmación es **falsa**. Aunque la entrada a un programa de agente y la entrada a la función del agente están relacionadas, no son exactamente la misma cosa.

Considera un agente de robot de limpieza:

* Entrada a la Función del Agente: Percepciones del entorno como "suciedad detectada en la esquina" o "pared frente al robot".
* Entrada al Programa del Agente: El programa puede recibir estas percepciones y también podría recibir datos adicionales como el estado de la batería del robot, el mapa del área, y otras métricas operativas.

**- Cada función de agente es implementable por alguna combinación de programa/máquina.**

R: La afirmación es **falsa**. No todas las funciones de agente son implementables por alguna combinación de programa y máquina.

Ejemplo de Implementación: Un agente que navega en un laberinto utilizando un algoritmo de búsqueda como A\* puede ser implementado en una máquina moderna con el programa adecuado. Aquí, la función del agente de encontrar la salida es computacionalmente factible.

Ejemplo de No Implementación: Considera una función de agente que requiere una decisión basada en información infinita o infinita capacidad de memoria. Por ejemplo, un agente que debe tomar decisiones basadas en todos los eventos futuros posibles en un entorno altamente complejo podría no ser implementable debido a limitaciones prácticas de la máquina y del programa.-

**- Suponga que un agente selecciona su acción uniformemente al azar del conjunto de**

**acciones posibles. Existe un entorno de tareas determinista en el que este agente es**

**racional.**

La afirmación es **falsa**. Un agente que selecciona su acción uniformemente al azar del conjunto de acciones posibles no puede ser racional en un entorno de tareas determinista, porque la racionalidad implica tomar decisiones que maximizan el rendimiento esperado basado en el conocimiento del entorno y los objetivos del agente.

**- Es posible que un agente dado sea perfectamente racional en dos entornos de tareas**

**distintos.**

La afirmación es **verdadera**. Es posible que un agente sea perfectamente racional en dos entornos de tareas distintos, siempre y cuando el agente esté diseñado para maximizar su rendimiento esperado de acuerdo con los objetivos y las percepciones disponibles en cada entorno específico.

Entorno 1:Un agente de navegación en un entorno de laberinto. Aquí, la racionalidad implica elegir la secuencia de movimientos que lleve al agente a la salida del laberinto.

Entorno 2: El mismo agente, pero en un entorno de planificación de rutas de transporte, donde la racionalidad implica encontrar la ruta más corta entre dos ubicaciones en una ciudad.

**- Todo agente es racional en un entorno no observable.**

La afirmación es **falsa**. No todo agente es racional en un entorno no observable. La racionalidad de un agente en un entorno depende de su capacidad para tomar decisiones que maximicen su rendimiento esperado basado en la información que tiene disponible. En un entorno no observable, donde el agente no puede percibir el estado completo del entorno, la racionalidad se vuelve mucho más complicada.

**- Un agente de póker perfectamente racional nunca pierde.**

La afirmación es falsa. Un agente de póker perfectamente racional no necesariamente nunca pierde. Aunque un agente racional en el póker tomará decisiones que maximicen su rendimiento esperado dado el conocimiento y la información disponible, esto no garantiza que nunca perderá.

**3. Para cada una de las siguientes actividades, dé una descripción PEAS del entorno de la tarea y caracterícelo en términos de las propiedades enumeradas en la Sección**

**- Jugar al fútbol.**

Respuesta: Jugar al fútbol es una tarea en un entorno parcialmente observable, ya que los jugadores no tienen información completa sobre todo el campo, influida por las posiciones y movimientos de los jugadores y el balón. Es estocástico, dado que los resultados de las acciones son inciertos y dependen de habilidades y condiciones externas, y secuencial, porque cada acción afecta directamente el estado del juego. El entorno es dinámico, ya que cambia constantemente con los movimientos de los jugadores y el balón, y continuo, pues las acciones y desplazamientos no están discretizados, aunque las reglas generan eventos puntuales. Además, es una tarea multiagente, con jugadores de ambos equipos interactuando de forma competitiva y cooperativa para anotar goles o defender. El rendimiento se mide por la cantidad de goles, minimización de faltas y efectividad en las jugadas, utilizando actuadores como movimientos corporales y comunicación, y sensores como visión, audición y propiocepción.

**- Explorar los océanos subterráneos de Titán (la luna más grande de Saturno).**

Respuesta: Explorar los océanos subterráneos de Titán se desarrolla en un entorno parcialmente observable, ya que la información está limitada por la falta de visibilidad directa y la dependencia de datos recopilados por sensores. Es estocástico, debido a la incertidumbre en las condiciones del entorno, como la composición química o la presión extrema, y secuencial, ya que cada paso en la exploración depende de decisiones y datos previos. El entorno es dinámico, dado que puede cambiar con fenómenos naturales, y continuo, ya que las mediciones y movimientos ocurren de forma fluida. Es una tarea de un solo agente, generalmente una sonda o robot autónomo que navega utilizando actuadores como propulsores y herramientas de perforación, y sensores como radares, cámaras y espectrómetros. El rendimiento se mide por la capacidad de recolectar datos científicos relevantes, evitar daños al equipo y cumplir objetivos de misión en condiciones extremas.

**- Comprar libros usados de IA en Internet.**

Respuesta: Comprar libros usados de IA en Internet se realiza en un entorno totalmente observable, ya que se puede acceder a toda la información relevante como precios, descripciones, condiciones y opiniones de otros usuarios. Es determinista, ya que las acciones como seleccionar un libro o realizar el pago tienen resultados predecibles, y secuencial, porque cada paso (buscar, comparar, comprar) afecta el siguiente. El entorno es estático, dado que no cambia mientras el usuario decide, y discreto, porque las acciones están bien definidas (elegir un libro, añadir al carrito, completar la compra). Es una tarea de un solo agente, donde el usuario interactúa con el sistema utilizando actuadores como clics y entradas de texto, y sensores como la vista para analizar las ofertas y leer reseñas. El rendimiento se mide por obtener el libro deseado al mejor precio y en las condiciones esperadas, con un proceso eficiente y seguro.

**- Jugar un partido de tenis.**

Respuesta: Jugar un partido de tenis se desarrolla en un entorno parcialmente observable, ya que el jugador no puede tener conocimiento total sobre las intenciones y estrategia del oponente, además de posibles limitaciones en la percepción del campo. Es estocástico, ya que el resultado de los golpes está sujeto a variabilidad en la precisión, velocidad y rebotes de la pelota, y secuencial, porque cada golpe afecta directamente el desarrollo del punto y del partido. El entorno es dinámico, con cambios continuos en la posición del oponente y la pelota, y continuo, debido al movimiento fluido del juego, aunque los puntos introducen cierta estructura discreta. Es una tarea de multiagente, con dos jugadores que interactúan competitivamente para ganar puntos. El rendimiento se mide por el número de puntos y juegos ganados, utilizando actuadores como movimientos corporales para correr, golpear y realizar saques, y sensores como la vista para rastrear la pelota y el oponente, además de la propiocepción para ajustar el cuerpo durante el juego.

**- Practicar tenis contra una pared.**

Respuesta: Practicar tenis contra una pared se lleva a cabo en un entorno totalmente observable, ya que el jugador tiene visibilidad completa del campo, la pelota y la pared. Es determinista, ya que el resultado de cada golpe es predecible, dependiendo de la fuerza y el ángulo del disparo, y secuencial, porque cada golpe influye en la trayectoria de la pelota para el siguiente. El entorno es dinámico, ya que cambia continuamente con el movimiento de la pelota, y continuo, debido a la naturaleza fluida del juego. Es una tarea de un solo agente, donde el jugador interactúa únicamente con la pared y la pelota. El rendimiento se mide por la capacidad de mantener un rally continuo y mejorar la precisión o técnica de los golpes, utilizando actuadores como movimientos corporales para golpear y moverse, y sensores como la vista para seguir la pelota y la propiocepción para ajustar el cuerpo en cada golpe.

**- Realizar un salto de altura.**

Respuesta: Realizar un salto de altura se desarrolla en un entorno totalmente observable, ya que el atleta tiene conocimiento completo de las condiciones del área de salto, la barra y su cuerpo. Es determinista, ya que el resultado depende de las acciones del atleta y las leyes físicas, y secuencial, porque cada fase del salto (carrera, impulso, vuelo y aterrizaje) afecta el éxito del intento. El entorno es estático, dado que no cambia mientras el salto está en curso, y continuo, debido a la fluidez de los movimientos. Es una tarea de un solo agente, donde el atleta interactúa únicamente con el suelo, la barra y el colchón. El rendimiento se mide por superar la altura de la barra sin derribarla, utilizando actuadores como movimientos corporales para correr, impulsarse y girar, y sensores como la vista para medir distancias y propiocepción para controlar la postura durante el salto.

**- Tejer un suéter.**

Respuesta: Tejer un suéter se realiza en un entorno totalmente observable, ya que el tejedor tiene control completo sobre las herramientas, el hilo y el diseño. Es determinista, ya que cada acción produce un resultado predecible en el tejido, y secuencial, porque cada punto depende de los puntos anteriores para formar el suéter correctamente. El entorno es estático, ya que no cambia mientras se realiza la tarea, y discreto, dado que el tejido se compone de puntos individuales que se realizan uno a la vez. Es una tarea de un solo agente, en la que el tejedor manipula agujas y hilo utilizando actuadores como movimientos manuales precisos, y sensores como la vista para supervisar el progreso y el tacto para percibir la tensión del hilo. El rendimiento se mide por la calidad, precisión y cumplimiento del diseño del suéter, además del uso eficiente del tiempo y los materiales.

**- Pujar por un artículo en una subasta.**

Respuesta: Pujar por un artículo en una subasta se lleva a cabo en un entorno parcialmente observable, ya que los participantes no tienen información completa sobre las estrategias, límites de oferta o intenciones de los otros postores. Es estocástico, ya que el resultado depende de las decisiones de otros participantes, y secuencial, ya que cada puja afecta el precio y las decisiones futuras. El entorno es dinámico, porque las pujas cambian constantemente durante la subasta, y discreto, ya que las acciones (realizar una puja) ocurren en etapas definidas. Es una tarea de multiagente, en la que múltiples postores compiten por el artículo. El rendimiento se mide por adquirir el artículo deseado al menor precio posible, utilizando actuadores como comandos verbales o clics para hacer las ofertas, y sensores como la vista y el oído para seguir el progreso de la subasta y evaluar las acciones de los demás.