Mundo de los bloques en DLV

Gabino Ruiz Ramirez
Benemérita Universidad autónoma de Puebla
FCC
202056050
gabino.ruizr@alumno.buap.mx

Jasmine Pérez Sánchez
Benemérita Universidad autónoma de Puebla
FCC
202073218
jasmine.perez@alumno.buap.mx

Jesús Huerta Aguilar
Benemérita Universidad autónoma de Puebla
FCC
202041509
jesus.huertaag@alumno.buap.mx

RESUMEN

El mundo de los bloques es un problema clásico en el campo de la inteligencia artificial (IA), que sirve para estudiar y probar algoritmos de planificación y manipulación. Este trabajo analiza tres archivos relacionados con el problema del mundo de los bloques: un archivo de hechos sobre las cajas, un archivo de planificación que define los fluentes y acciones permitidas, y un archivo de plan inicial y objetivo. Los archivos están diseñados para resolver una configuración particular del problema, en la que tres bloques (a, b y c) deben ser organizados siguiendo una secuencia de acciones lógicas. El análisis expone la lógica utilizada, el uso de condiciones para la ejecución de acciones, las restricciones de concurrencia y el estado inicial y final del entorno simulado. Este sistema proporciona una demostración efectiva del uso de la lógica declarativa y algoritmos de planificación en entornos de IA simplificados, mostrando su utilidad en la resolución de problemas secuenciales.

PALABRAS CLAVE

Mundo de los bloques, Planificación automática, Lógica declarativa, IA simbólica, Manipulación de objetos, Restricciones de concurrencia.

ABSTRACT

The blocks world is a classic problem in the field of artificial intelligence (AI), used to study and test planning and manipulation algorithms. This paper analyzes three files related to the blocks world problem: a fact file about the boxes, a planning file that defines the fluents and permitted actions, and a file containing the initial and goal plan. These files are designed to solve a particular configuration of the problem, in which three blocks (a, b, and c) must be organized following a sequence of logical actions. The analysis explains the logic used, the application of conditions for action execution, concurrency constraints, and the initial and final state of the simulated environment. This system provides an effective demonstration of the use of declarative logic and planning algorithms in simplified AI environments, showing their utility in solving sequential problems.

KEYWORDS

Blocks world, Automated planning, Declarative logic, Symbolic AI, Object manipulation, Concurrency constraints.

INTRODUCCIÓN

El mundo de los bloques es un paradigma fundamental en el desarrollo de sistemas de inteligencia artificial (IA) enfocados en la planificación automática, lógica de predicados y manipulación de objetos. Introducido en la década de 1970, este modelo ha sido utilizado para estudiar cómo un agente inteligente puede planificar secuencias de acciones, respetando las restricciones físicas y lógicas de un entorno. El problema se estructura de manera que un conjunto de bloques apilables debe ser reorganizado, partiendo de una configuración inicial hasta una disposición objetivo, respetando reglas predefinidas sobre qué acciones son posibles en cada momento.

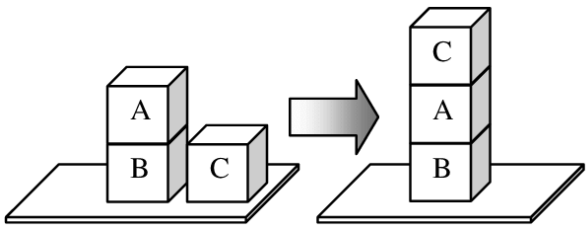

Ilustración 1: Mundo de los bloques

En este ensayo, se examinan tres archivos relacionados con la implementación de este problema en un sistema de planificación lógico. El archivo file1.lp define los bloques y sus posibles ubicaciones en el entorno; file2.plan establece los fluentes (condiciones del entorno) y las acciones permitidas, incluyendo restricciones para su ejecución, y file3.plan define el estado inicial del entorno y el objetivo final a alcanzar. El análisis de estos archivos permite demostrar cómo un sistema lógico puede modelar la toma de decisiones en entornos controlados, simulando de manera eficiente el comportamiento de un agente inteligente en un mundo simplificado.

METODOLOGIA

Para resolver el problema del mundo de los bloques y su correspondiente secuencia de acciones, hemos implementado una solución basada en lógica declarativa y planificación automática, dividida en tres archivos clave. Estos archivos definen los elementos del entorno, las reglas y restricciones para las acciones posibles, y el estado inicial y objetivo del sistema. A continuación, se explican en detalle las funciones de cada archivo, junto con la metodología empleada para lograr una planificación secuencial efectiva y el cumplimiento de las condiciones impuestas.

FILE1.LP: DEFINICIÓN DEL ENTORNO Y DE LOS OBJETOS

El archivo file1.lp define los objetos con los que trabajará el agente en el entorno. En este caso, se definen tres bloques o "cajas" que corresponden a las letras a, b, y c, además de la **mesa** como un lugar disponible para colocar los bloques.

```
caja(a). caja(b). caja(c).
lugar(X):-caja(X).
lugar(mesa):-true.
```

- **caja(a), caja(b), caja(c):** Estas declaraciones indican que a, b y c son bloques que el agente puede manipular.
- **lugar(X):-caja(X):** Esta regla establece que cualquier objeto definido como caja también es un **lugar** sobre el cual pueden colocarse otros bloques.
- **lugar(mesa):-true:** La mesa es un lugar válido por defecto (siempre verdadero) donde pueden colocarse bloques.

Este archivo establece la base de conocimiento necesaria para que el agente reconozca los objetos y lugares en el entorno, permitiendo así el razonamiento sobre los posibles movimientos de los bloques.

FILE2.PLAN: DEFINICIÓN DE FLUENTES, ACCIONES Y RESTRICCIONES

El archivo file2.plan es el núcleo de la lógica del sistema. Define los **fluentes**, que son condiciones o estados del entorno, así como las **acciones** que puede ejecutar el agente, junto con sus respectivas restricciones. Además, incluye las reglas de causalidad que describen cómo cambian los estados en el entorno como resultado de las acciones.

Fluentes

Los fluentes representan estados lógicos del entorno, como la posición de los bloques o el estado de la garra (arriba o abajo):

```
fluents:
     sobre(B,L) requires caja(B), lugar(L).
     ocupada(L) requires lugar(L).
     sosteniendo(B) requires caja(B).
     garra_arriba.
     garra_abajo.
```

- **sobre(B,L):** Indica que el bloque B está sobre el lugar L, que puede ser otro bloque o la mesa.
- **ocupada(L):** Indica que el lugar L (otro bloque o la mesa) está ocupado, lo que implica que no se pueden colocar más bloques en él.
- **sosteniendo(B):** Representa que el agente está sosteniendo el bloque B.
- **garra_arriba/garra_abajo:** Representan el estado de la garra del agente, que solo puede estar en uno de estos dos estados.

Acciones

Las acciones que puede ejecutar el agente son fundamentales para la planificación. Estas acciones están sujetas a condiciones, representadas en las reglas que definen cuándo y cómo pueden ser ejecutadas:

```
actions:
     move(B,L) requires caja(B), lugar(L).
     subir requires garra_abajo.
     bajar requires garra_arriba.
     coger(B) requires caja(B), garra_abajo, not
sosteniendo(AlgunB), sobre(B,L), not ocupada(B).
     soltar(B,L) requires lugar(L), garra_abajo,
sosteniendo(B), not ocupada(L).
```

- **move(B,L):** Mueve el bloque B al lugar L, sujeto a que B sea una caja y L sea un lugar válido.
- **subir/bajar:** Suben o bajan la garra, según su estado actual.
- **coger(B):** El agente puede coger un bloque B si la garra está abajo, no está sosteniendo otro bloque, y B está sobre un lugar L no ocupado.
- **soltar(B,L):** El agente puede soltar el bloque B sobre el lugar L si no está ocupado.

Reglas de causalidad y restricciones de concurrencia

Se definen reglas para asegurar que las acciones cambien el estado del entorno de forma coherente y para evitar que múltiples acciones ocurran simultáneamente.

```
always:
     executable move(B,L) if not ocupada(B), not
ocupada(L), B <> L.
     executable subir if garra_abajo.
     executable bajar if garra_arriba.
     executable coger(B) if sobre(B,L), not
ocupada(B), not sosteniendo(B), garra_abajo.
     executable soltar(B,L) if sosteniendo(B), not
ocupada(L), garra_abajo.


     inertial sobre(B,L).
     caused ocupada(L) if sobre(B,L), lugar(L).
     caused sobre(B,L) after move(B,L).
     caused -sobre(B,L1) after move(B,L), sobre(B,L1),
L <> L1.
```

file3.plan: Estado inicial y objetivo
El archivo file3.plan define el **estado inicial** del entorno y el **estado objetivo** que el agente debe alcanzar.

```
initially:
     sobre(a,mesa).  sobre(b,mesa).  sobre(c,mesa).
     garra_abajo.


goal:
     sobre(a,b),  sobre(b,c),  sobre(c,mesa)  ?  (2)
```

- **Estado inicial:** Los bloques a, b y c están sobre la mesa, y la garra comienza en el estado "abajo".
- **Objetivo:** El agente debe organizar los bloques de manera que a esté sobre b, b esté sobre c, y c esté sobre la mesa.

PROCESO DE PLANIFICACIÓN
La metodología consiste en que el sistema **evalúa el estado inicial**, selecciona las acciones permitidas, y aplica las reglas de causalidad para actualizar los fluentes hasta que se cumpla el objetivo. Cada acción cambia el estado del entorno, afectando la posición de los bloques y el estado de la garra, lo que permite una secuencia lógica de eventos.
En resumen, este sistema sigue una metodología basada en reglas lógicas para modelar tanto el entorno como las acciones, garantizando que el agente siga una secuencia de movimientos factibles y eficientes para alcanzar la meta final en el mundo de los bloques.

RESULTADOS

Tras la ejecución del sistema descrito en los archivos de planificación (file1.lp, file2.plan y file3.plan), los resultados obtenidos muestran cómo el agente sigue una secuencia lógica de acciones para resolver el problema del **mundo de los bloques**, partiendo del estado inicial definido hasta alcanzar el estado objetivo. El sistema genera una planificación secuencial y coherente, asegurando el cumplimiento de las restricciones impuestas sobre las acciones y los fluentes del entorno.

EJECUCIÓN DEL PLAN
El proceso de planificación empieza con la **configuración inicial**: todos los bloques (a, b y c) se encuentran sobre la mesa y la garra está abajo. A partir de este estado inicial, el agente sigue una serie de pasos para mover los bloques en el orden necesario para alcanzar la disposición final deseada. Los siguientes son los pasos principales que el agente sigue:

1. **Coger el bloque a:**
   - **Acción:** El agente baja la garra (bajar) para estar en posición de coger el bloque a, que está libre en la mesa.
   - **Condiciones cumplidas:** garra_abajo, sobre(a,mesa), not ocupada(a), not sosteniendo(otro bloque).
   - **Resultado:** El bloque a es levantado, lo que activa el fluente sosteniendo(a).

2. **Colocar el bloque a sobre el bloque b:**
   - **Acción:** El agente mueve el bloque a sobre el bloque b con la acción soltar(a,b).
   - **Condiciones cumplidas:** sosteniendo(a), not ocupada(b), garra_abajo.
   - **Resultado:** Se actualizan los fluentes a sobre(a,b) y ocupada(b).

3. **Coger el bloque b:**
   - **Acción:** El agente baja la garra y coge el bloque b, que está libre en la mesa.
   - **Condiciones cumplidas:** garra_abajo, sobre(b,mesa), not ocupada(b).
   - **Resultado:** El bloque b es sostenido por la garra, activando sosteniendo(b).

4. **Colocar el bloque b sobre el bloque c:**
   - **Acción:** El bloque b es colocado sobre el bloque c usando la acción soltar(b,c).
   - **Condiciones cumplidas:** sosteniendo(b), not ocupada(c), garra_abajo.
   - **Resultado:** Se actualiza el fluente a sobre(b,c) y ocupada(c).

5. **Coger el bloque c:**
   - **Acción:** El agente coge el bloque c de la mesa usando coger(c).
   - **Condiciones cumplidas:** garra_abajo, sobre(c,mesa), not ocupada(c).
   - **Resultado:** El bloque c es sostenido por la garra, activando sosteniendo(c).

6. **Colocar el bloque c sobre la mesa:**
   - **Acción:** El agente suelta el bloque c en la mesa con la acción soltar(c,mesa).
   - **Condiciones cumplidas:** sosteniendo(c), not ocupada(mesa), garra_abajo.
   - **Resultado:** El bloque c está sobre la mesa, completando la configuración deseada.

2. Validación del estado objetivo
Después de ejecutar la secuencia de acciones, el sistema comprueba que el estado objetivo se ha alcanzado correctamente. Los fluentes resultantes son:

- sobre(a,b): El bloque a está sobre el bloque b.
- sobre(b,c): El bloque b está sobre el bloque c.
- sobre(c,mesa): El bloque c está sobre la mesa.

Este es exactamente el estado deseado según la declaración del objetivo en file3.plan:

```
goal:
   sobre(a,b), sobre(b,c), sobre(c,mesa).
```

3. Secuencia de acciones ejecutadas

La planificación del agente implica la ejecución de las siguientes acciones, en el orden correcto y cumpliendo con las restricciones de concurrencia y las condiciones necesarias en cada paso:

1. **bajar**
2. **coger(a)**
3. **subir**
4. **bajar**
5. **soltar(a,b)**
6. **bajar**
7. **coger(b)**
8. **subir**
9. **bajar**
10. **soltar(b,c)**
11. **bajar**
12. **coger(c)**
13. **subir**
14. **bajar**
15. **soltar(c,mesa)**

4. Resultados generales del sistema

- **Coherencia y precisión:** El sistema de planificación ejecuta una secuencia lógica de acciones que respeta todas las reglas impuestas por los fluentes y las restricciones definidas. En ningún momento se intentan ejecutar acciones simultáneamente, gracias a la regla noConcurrency.
- **Optimización:** Aunque este ejemplo se basa en un número reducido de bloques y acciones, la secuencia de acciones es eficiente, ya que no se realizan movimientos innecesarios y cada bloque es movido una sola vez hasta alcanzar su posición final.
- **Correctitud:** El agente logra reorganizar los bloques de acuerdo con la especificación del problema, alcanzando el estado final deseado. Los fluentes resultantes confirman que la planificación fue exitosa, logrando el objetivo propuesto de apilar a sobre b, b sobre c, y c sobre la mesa.

Ilustración 2: ejecucion en poweshell

CONCLUSIONES

El análisis del problema del mundo de los bloques utilizando una metodología basada en **lógica declarativa** y **planificación automática** demuestra que este enfoque es eficaz para resolver problemas secuenciales en entornos controlados. A través de la definición de fluentes y reglas precisas, se logró que el agente manipulara correctamente los bloques, alcanzando el estado objetivo de manera eficiente.

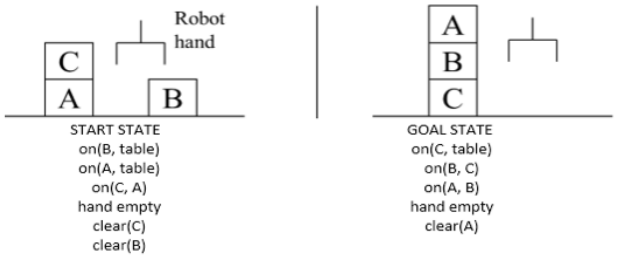

Ilustración 3: estado de inicio y meta

El sistema propuesto garantiza coherencia en las acciones, asegurando que cada movimiento respete las condiciones y restricciones impuestas. A pesar de ser un escenario simplificado, este modelo es un excelente ejemplo de cómo la **IA simbólica** y las técnicas de planificación pueden aplicarse a problemas de manipulación física. Si bien la complejidad podría aumentar en problemas de mayor escala, este ejercicio demuestra la capacidad de los algoritmos de planificación para guiar agentes en la resolución de tareas complejas, proporcionando una base sólida para el estudio y desarrollo de sistemas más avanzados en inteligencia artificial.

AGRADECIMIENTOS

Quiero expresar mi más sincero agradecimiento al profesor Fernando Zacarias Flores, cuya orientación y apoyo han sido fundamentales en el desarrollo y comprensión del problema del **mundo de los bloques** y su implementación en **planificación automática**. Su pasión por la enseñanza y su compromiso con la educación en **inteligencia artificial** y **lógica declarativa** han sido invaluables para la realización de este proyecto.

El profesor Zacarias Flores ha proporcionado una guía experta en las técnicas necesarias para desglosar el problema del mundo de los bloques en sus componentes fundamentales: fluentes, acciones y restricciones. Su habilidad para explicar conceptos complejos, como la secuenciación lógica de acciones y la manipulación de objetos, ha sido crucial para entender cómo aplicar estos principios de forma efectiva en la planificación de IA, permitiéndonos resolver problemas secuenciales con precisión.

Además, la retroalimentación constructiva y el interés genuino del profesor por el proyecto han sido clave para mejorar nuestra implementación. Su apoyo ha permitido una comprensión más profunda de los desafíos técnicos, como la gestión de restricciones

de concurrencia y el manejo de estados inerciales, facilitando el diseño de un agente que resuelve de manera óptima la configuración del mundo de los bloques.

El profesor Zacarias también ha fomentado un ambiente de creatividad y experimentación, animándonos a explorar nuevas formas de **modelar sistemas lógicos** y encontrar soluciones innovadoras en el campo de la **IA simbólica**. Su compromiso con la formación de futuros especialistas en inteligencia artificial ha sido una fuente constante de inspiración y motivación.

Agradezco profundamente el tiempo que el profesor Zacarias ha dedicado en guiarnos y compartir su vasta experiencia en **planificación automática** y lógica computacional, lo cual ha sido esencial para el éxito de este proyecto. Su influencia ha dejado una huella duradera en mi desarrollo académico y profesional, y su guía seguirá siendo valiosa en mis futuros emprendimientos en el campo de la inteligencia artificial.

Finalmente, quiero reconocer el esfuerzo y la dedicación de todos los que han colaborado en este proyecto, así como la comunidad académica que promueve el avance del conocimiento en **IA y planificación**. La colaboración y el intercambio de ideas han sido esenciales para lograr resultados satisfactorios en este ámbito en constante evolución.

REFERENCIAS

[1] Brownlee, J. (2019). *Deep Learning for Natural Language Processing: Develop Deep Learning Models for your Natural Language Problems*. Machine Learning Mastery

[2] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems* (pp. 8024-8035).

[3] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Rush, A. M. (2020). Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (pp. 38-45).