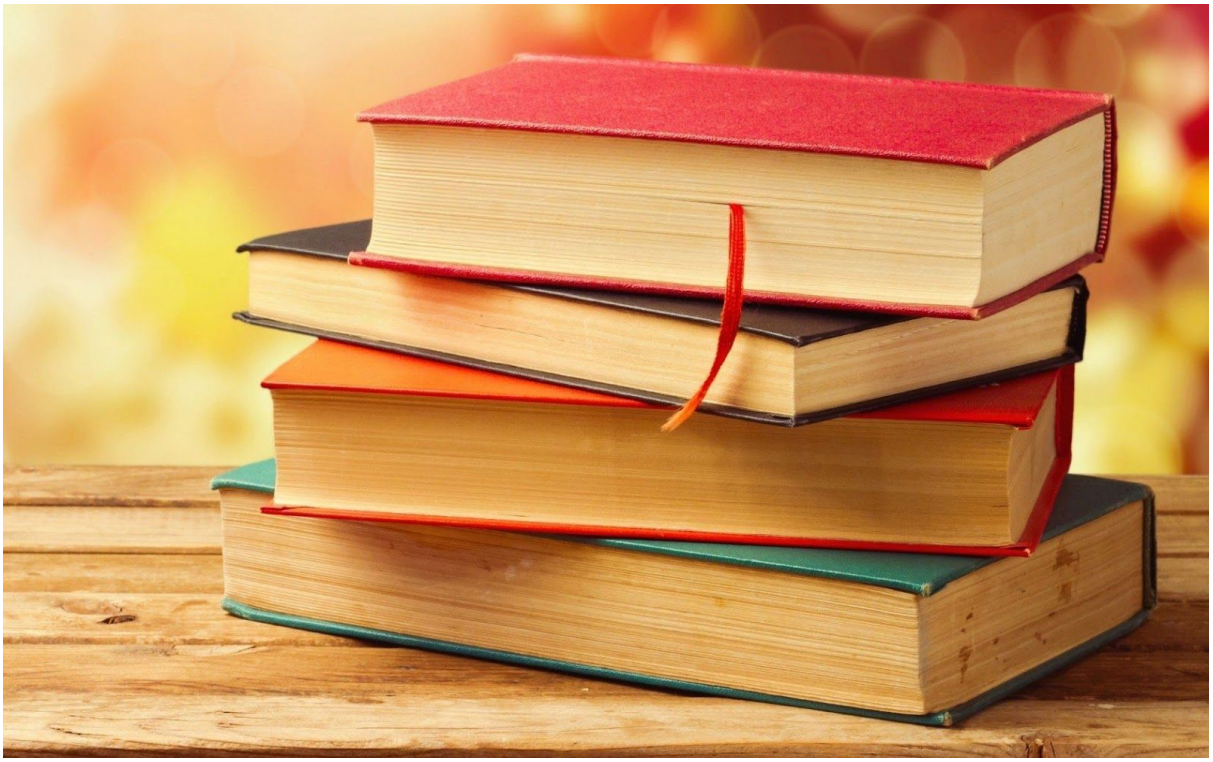
PRÁCTICA 3

INTELIGENCIA ARTIFICIAL

Planificador de libros



Isaac Muñoz Busto, Ignasi Sant Albors, Pablo Fonoll Soto

Índice

1. Modelización

1.1. Dominio

- **Variables**

  libro: objeto que identifica un libro.
  mes: objeto que indica un mes.

- **Funciones**

  (ordenMes ?m - mes) : número que tiene un mes en concreto, para poder establecer un orden.
  (ultimoMes ?l - libro) : el último mes que se le ha asignado a un predecesor de este libro, para poder cumplir el orden entre libros con predecesores.
  (predecesores ?l - libro) : la cantidad de predecesores que tiene cada libro, para poder cumplir el orden entre libros con predecesores.
  (predecesoresAsignados ? - libro) : la cantidad de predecesores de cada libro que ya se ha asignado a un día.

- **Predicados**

  (predecesor ?x - libro ?y - libro): indica si el libro x es anterior al libro y.
  (leido ?x - libro): indica si el libro x se ha leido;
  (leer ?x - libro): indica si el libro x se ha de leer.
  (libroAsignado ?l - libro) : indica si un libro ya tiene asignado un mes para su lectura.

- **Acciones**

  leer_predecesor(x, y): si x es predecede y y x no se ha leido pero el usuario quiere leer y , marca que se debe leer x.
  senalar_libro(x): señala que x será el siguiente libro a leer.
  ordenar_libros(m, x): para cada combinación de mes m y libro x busca cómo asignar los días para que se cumplan las restricciones de predecesores de cada libro.

1.2. Problemas

1.2.1. Modelo Básico

En el plan de lectura todos los libros tienen 0 o 1 predecesores y ningún contenido es paralelo. El planificador tiene la función de recomendar los libros deseados. En este modelo solo hay un “type” por libro, ya que solo nos preocuparemos de encontrar el libro a mostrar. Hemos creado el predicado (predecesor ?x - contenido ?y - contenido), sobre este predicado nos aseguramos que, si no ha leído un libro que tiene relación predecesor con un libro que se quiere leer, el resultado mostrará el libro deseado y su predecesor.

2. Desarrollo

Hemos decidido hacer un diseño incremental basado en prototipos. Hemos empezado implementando el modelo básico y, al finalizarlo, hemos implementado las demás extensiones. A medida que íbamos programando los nuevos modelos, hemos ido añadiendo nuevos elementos, por ejemplo, en la extensión 3 hemos añadido el número de páginas de los libros.
Hemos creado un pequeño programa en c++ para generar diferentes juegos de prueba y probarlos en la PDDL.

3. Problemas de prueba

Prueba 1

Input:

```
(define (problem prova0) (:domain planificador)
      (:objects l0 l1 l2 - libro)
(:init
(predecesor l1 l2)
(leer l2)
)

(:goal (and (forall (?x - libro) (not (leer ?x))))))
```

Output:

```
step    0: LEER_PREDECESOR L1 L2
        1: LEER_LIBRO L1
        2: LEER_LIBRO L2
```

Justificación: Este juego de pruebas para el modelo básico lo hemos escogido porque, al ser el libro 1 predecesor del libro 2, el programa nos tiene que mostrar tanto el libro 1 como el libro 2. Puesto que, pese a solo querer leer el libro 2, para hacerlo es necesario leer también el libro 1, aunque no hayamos explicitado que queremos leer el libro 1.

Prueba 2

Input:

```
(define (problem prova0) (:domain planificador)
      (:objects l0 l1 l2 - libro)
(:init
(predecesor l1 l2)
(leer l2)
(leido l1)
```

```
)
(:goal (and (forall (?x - libro) (not (leer ?x))))))
```

Output:

```
step
        1: LEER_LIBRO L2
```

Justificación: Este juego de pruebas es muy parecido al anterior, pero con la diferencia de que esta vez sí hemos leído el libro 1, por lo tanto la salida tiene que ser exclusivamente el libro 2, puesto que el libro 1, pese a ser precedente, ya lo hemos leído.

Prueba 3

Input:

```
(define (problem prova0) (:domain planificador)
      (:objects l0 l1 l2 - libro)
(:init
(predecesor l1 l2) // podriamos poner (predecesor l1 l3) y también funciona
(predecesor l2 l3)
(leer l3)
)

(:goal (and (forall (?x - libro) (not (leer ?x))))))
```

Output:

```
step
      0: LEER_PREDECESOR L2 L3
      1: LEER_PREDECESOR L1 L2
      2: LEER_LIBRO L1
      3: LEER_LIBRO L2
      4: LEER_LIBRO L3
```

Justificación: Este juego de pruebas comprueba que un libro pueda tener varios precedentes , en este caso hemos hecho que el libro 3 tenga como precedentes los libros 1 y 2. Por lo tanto el sistem nos planifica primero la lectura de los libros 1 y 2 y finalmente el 3. Se ha elegido este juego de prueba para hacer testing de varios predecesores.