Práctica de Planificación: Redflix

Inteligencia Artificial

Xinxiang Zhou
Pablo Fondevilla
Carlos Rosales

Curso 2024-25 - Q1

Índice

1. El Problema

Redflix es una plataforma que ofrece a sus usuarios una amplia variedad de contenidos audiovisuales, desde series hasta películas. La plataforma quiere desarrollar una aplicación que permita a sus usuarios planificar la visualización de su contenido favorito.

Esta nueva herramienta deberá ser capaz de tener en cuenta los contenidos que los usuarios ya han visto y aquellos que quieren ver, para ofrecer un plan de visualización diario. Este plan indicará qué contenido ver y en qué orden. Las principales restricciones que deberá considerar la nueva aplicación son:

- **Contenidos Predecesores**: Las series o películas predecesoras son aquellos contenidos que deben haberse visto antes de visionar a un determinado contenido, ya que son importantes para entender la trama.
- **Contenidos Paralelos**: Un contenido paralelo es una historia que se desarrolla dentro del mismo universo ficticio que otra película o serie, pero que se narra en producciones diferentes.

A continuación veremos con más detalle cada uno de los problemas de esta práctica.

1.1. Nivel básico

En el nivel básico solo pueden existir contenidos con 0 o 1 predecesores y ningún paralelo.

- **Estado inicial**: Se indica los contenidos que existen como objetos, se definen los predecesores y los contenidos que se quiere ver.
- **Estado final**: Se muestra una planificación de los contenidos que el usuario quería ver.

1.2. Extensión 1

En esta extensión, los contenidos pueden tener de 0 a N predecesores y ningún paralelo. Y para todo contenido que pertenece al plan, todos sus contenidos predecesores pertenecen al plan y están en días anteriores.

- **Estado inicial**: Se especifican los contenidos y los días disponibles como objetos, se declaran las relaciones de predecesores entre los contenidos, y se establece un orden entre los días.
- **Estado final**: Se muestra la planificación necesaria para que el usuario pueda marcar como vistos los contenidos especificados en el *goal*. La planificación muestra que contenido ver cada día respetando el orden de los predecesores. Si un contenido de la planificación tiene al menos un predecesor, entonces el predecesor también estará en la planificación.

1.3. Extensión 2

La segunda extensión es similar a la primera, pero esta vez debemos tener en cuenta los contenidos paralelos. Los contenidos pueden tener de 0 a M contenidos paralelos.

- **Estado inicial**: Se especifican los contenidos y los días disponibles como objetos, se declaran las relaciones de predecesores y paralelismo entre los contenidos, y se establece la relación de *día siguiente* y *día anterior* entre los días.
- **Estado final**: Se muestra la planificación necesaria para que el usuario pueda marcar como vistos los contenidos especificados en el *goal*. La planificación muestra que contenido ver cada día respetando el orden de los predecesores y los contenidos paralelos. Si un contenido de la planificación tiene al menos un contenido predecesor o paralelo, entonces el predecesor o el paralelo también estarán en la planificación.

1.4. Extensión 3

La tercera extensión toma como base la segunda. Este nivel controla que no se coloquen más de 3 contenidos al día.

- **Estado inicial**: Se especifican los contenidos y los días disponibles como objetos, se declaran las relaciones de predecesores y paralelismo entre los contenidos, y se establece la relación de *día siguiente* y *día anterior* entre los días. Por último, inicializamos a 0 el número de contenidos asignados para todos los días.
- **Estado final**: Se muestra la planificación necesaria para que el usuario pueda marcar como vistos los contenidos especificados en el *goal*. La planificación muestra que contenido ver cada día respetando el orden de los predecesores, los contenidos paralelos y el número máximo de contenidos por día. Si un contenido de la planificación tiene al menos un contenido predecesor o paralelo, entonces el predecesor o el paralelo también estarán en la planificación.

1.5. Extensión 4

En la última extensión, a cada contenido se le asocia una duración, el planificador deberá generar un plan que no supere los 200 minutos al día.

- **Estado inicial**: Se especifican los contenidos y los días disponibles como objetos, se declaran las relaciones de predecesores y paralelismo entre los contenidos, y se establece la relación de *día siguiente* y *día anterior* entre los días. Por último, inicializamos la duración de cada contenido y asignamos 0 minutos visualizados a cada día.

- **Estado final**: Se muestra la planificación necesaria para que el usuario pueda marcar como vistos los contenidos especificados en el *goal*. La planificación muestra que contenido ver cada día respetando el orden de los predecesores, los contenidos paralelos y el número máximo de minutos por día. Si un contenido de la planificación tiene al menos un contenido predecesor o paralelo, entonces el predecesor o el paralelo también estarán en la planificación.

2. Dominio

Antes de entrar en detalle con cada uno de los dominios, nos gustaría justificar la elección del único operador que hemos utilizado en esta práctica.

Como podemos observar en los siguientes apartados, cada extensión posee un solo operador. En un principio, pensamos en incluir más de uno, como los operadores `seleccionar-contenido`, `asignar-dia` o `buscar-contenido`. Sin embargo, preferimos mantener un único operador por simplicidad, ya que la acción del usuario no cambia, son las restricciones las que la modifican. Por esta razón, decidimos trasladar la lógica de las restricciones a la precondición del operador. De esta forma, nos aseguramos de que, en cada momento, el operador sea el adecuado y pueda abarcar todos los casos.

2.1. Nivel básico

En el nivel básico, no existen restricciones relacionadas con los días. Por ello, hemos optado por planificar los contenidos para un solo día. Por esta razón, no hemos definido `dia` como un tipo en el dominio. Para asegurar que el planificador cumpla estas restricciones, hemos definido los siguientes tipos, predicados y acciones:

Tipos:

- `contenido`: Representa las películas y los episodios de series televisivas.

Predicados:

- `(predecesor ?pre - contenido ?post - contenido)`: Indica que el contenido `?pre` es predecesor del contenido `?post`. Por lo tanto, `?pre` debe visualizarse en un día anterior al planificado para ver `?post`. Necesario para modelar las dependencias entre los contenidos.
- `(ha-visto ?c - contenido)`: Indica que el contenido `?c` ya ha sido visualizado por el usuario. Necesario para llevar un seguimiento de los contenidos visualizados y evitar volver a verlos.
- `(quiere-ver ?C - contenido)`: Indica el contenido `?c` que el usuario quiere ver. Es el objetivo de la planificación para el usuario.

Acciones:

- `(ver (parameters (?c - contenido)))`: Esta acción asegura que un contenido no puede ser visto si ya ha sido visto, y evita que este sea visualizado sin cumplir con las dependencias de precedencia. El efecto de esta acción marca el contenido `?c` como visto.

2.2. Extensión 1

En este dominio añadimos el tipo `dia` para gestionar todos los predecesores posibles.

Tipos:

- `contenido`: Representa las películas y los episodios de series televisivas.
- `dia`: Representa los días posibles para planificar los contenidos.

Predicados:

- `(predecesor ?pre - contenido ?post - contenido)`: Indica que el contenido `?pre` es predecesor del contenido `?post`. Por lo tanto, `?pre` debe visualizarse en un día anterior al planificado para ver `?post`. Necesario para modelar las dependencias entre los contenidos.
- `(ha-visto ?c - contenido)`: Indica que el contenido `?c` ya ha sido visualizado por el usuario. Necesario para llevar un seguimiento de los contenidos visualizados y evitar volver a verlos.
- `(dia-anterior ?prev - dia ?next - dia)`: Indica que el día `?prev` precede cronológicamente al día `?next`. Necesario mantener la coherencia temporal del dominio.
- `(asignado ?c - contenido ?d - dia)`: Indica que el contenido `?c` está asignado al día `?d`. Necesario para planificar en qué día se debe visualizar cada contenido.

Acciones:

- `(ver (parameters (?c - contenido ?dia - dia)))`: Esta acción asegura que un contenido no puede ser visto si ya ha sido visto, y evita que este sea visualizado sin cumplir con las dependencias de precedencia. El efecto de esta acción marca el contenido `?c` como visto y lo asigna al día `?dia`.

2.3. Extensión 2

En la segunda extensión añadimos los predicados necesarios para gestionar los contenidos paralelos y predecesores.

Tipos:

- `contenido`: Representa las películas y los episodios de series televisivas.
- `dia`: Representa los días posibles para planificar los contenidos.

Predicados:

- `(predecesor ?pre - contenido ?post - contenido)`: Indica que el contenido `?pre` es predecesor del contenido `?post`. Por lo tanto, `?pre` debe visualizarse en un día anterior al planificado para ver `?post`. Necesario para modelar las dependencias entre los contenidos.
- `(paralelo ?c1 - contenido ?c2 - contenido)`: Indica que el contenido `?c1` es paralelo con el contenido `?c2`. Necesario para modelar las dependencias entre los contenidos.
- `(ha-visto ?c - contenido)`: Indica que el contenido `?c` ya ha sido visualizado por el usuario. Necesario para llevar un seguimiento de los contenidos visualizados y evitar volver a verlos.
- `(asignado ?c - contenido ?d - dia)`: Indica que el contenido `?c` está asignado al día `?d`. Necesario para planificar en qué día se debe visualizar cada contenido.
- `(dia-anterior ?prev - dia ?next - dia)`: Indica que el día `?prev` precede cronológicamente al día `?next`. Necesario mantener la coherencia temporal del dominio.
- `(dia-siguiente ?next - dia ?prev - dia)`: Indica que el día `?next` es el siguiente cronológicamente al día `?prev`. Necesario para asignar un contenido paralelo a los días adyacentes. Si un contenido `c1` es paralelo a otro, este último también será paralelo a `c1`. Por lo tanto, tanto `c1` como `c2` pueden asignarse después del día en que se planificó el otro contenido.

Acciones:

- `(ver (parameters (?c - contenido ?dia - dia)))`: Esta acción asegura que un contenido no puede ser visto si ya ha sido visto, y evita que este sea visualizado sin cumplir con las dependencias de precedencia. Además, si un contenido es paralelo a otro, ese otro contenido debe ser asignado al mismo día o a días consecutivos. El efecto de esta acción marca el contenido `?c` como visto y lo asigna al día `?dia`.

2.4. Extensión 3

En este dominio se añaden fluentes para poder utilizar funciones y operaciones aritméticas.

Tipos:

- `contenido`: Representa las películas y los episodios de series televisivas.
- `dia`: Representa los días posibles para planificar los contenidos.

Predicados:

- `(predecesor ?pre - contenido ?post - contenido)`: Indica que el contenido `?pre` es predecesor del contenido `?post`. Por lo tanto, `?pre` debe visualizarse en un día anterior al planificado para ver `?post`. Necesario para modelar las dependencias entre los contenidos.
- `(paralelo ?c1 - contenido ?c2 - contenido)`: Indica que el contenido `?c1` es paralelo con el contenido `?c2`. Necesario para modelar las dependencias entre los contenidos.
- `(ha-visto ?c - contenido)`: Indica que el contenido `?c` ya ha sido visualizado por el usuario. Necesario para llevar un seguimiento de los contenidos visualizados y evitar volver a verlos.
- `(asignado ?c - contenido ?d - dia)`: Indica que el contenido `?c` está asignado al día `?d`. Necesario para planificar en qué día se debe visualizar cada contenido.
- `(dia-anterior ?prev - dia ?next - dia)`: Indica que el día `?prev` precede cronológicamente al día `?next`. Necesario mantener la coherencia temporal del dominio.
- `(dia-siguiente ?next - dia ?prev - dia)`: Indica que el día `?next` es el siguiente cronológicamente al día `?prev`. Necesario para asignar un contenido paralelo a los días adyacentes. Si un contenido `c1` es paralelo a otro, este último también será paralelo a `c1`. Por lo tanto, tanto `c1` como `c2` pueden asignarse después del día en que se planificó el otro contenido.

Funciones:

- `(num-asignados ?d - dia)`: Contador del número de contenidos asignados al día `?d`.

Acciones:

- `(ver (parameters (?c - contenido ?dia - dia))`: Esta acción asegura que un contenido no puede ser visto si ya ha sido visto, y garantiza que se cumplan las relaciones de precedencia y paralelismo. Por último, comprueba que el dia `?dia` tenga menos de 3 contenidos asignados. El efecto de esta acción marca el contenido `?c` como visto, lo asigna al día `?dia` y aumenta en 1 el contador del día `?dia`.

2.5. Extensión 4

En la extensión 4, definimos la duración de los contenidos (en minutos) mediante funciones.

Tipos:

- `contenido`: Representa las películas y los episodios de series televisivas.
- `dia`: Representa los días posibles para planificar los contenidos.

Predicados:

- `(predecesor ?pre - contenido ?post - contenido)`: Indica que el contenido `?pre` es predecesor del contenido `?post`. Por lo tanto, `?pre` debe visualizarse en un día anterior al planificado para ver `?post`. Necesario para modelar las dependencias entre los contenidos.
- `(paralelo ?c1 - contenido ?c2 - contenido)`: Indica que el contenido `?c1` es paralelo con el contenido `?c2`. Necesario para modelar las dependencias entre los contenidos.
- `(ha-visto ?c - contenido)`: Indica que el contenido `?c` ya ha sido visualizado por el usuario. Necesario para llevar un seguimiento de los contenidos visualizados y evitar volver a verlos.
- `(asignado ?c - contenido ?d - dia)`: Indica que el contenido `?c` está asignado al día `?d`. Necesario para planificar en qué día se debe visualizar cada contenido.
- `(dia-anterior ?prev - dia ?next - dia)`: Indica que el día `?prev` precede cronológicamente al día `?next`. Necesario mantener la coherencia temporal del dominio.
- `(dia-siguiente ?next - dia ?prev - dia)`: Indica que el día `?next` es el siguiente cronológicamente al día `?prev`. Necesario para asignar un contenido paralelo a los días adyacentes. Si un contenido `c1` es paralelo a otro, este último también será paralelo a `c1`. Por lo tanto, tanto `c1` como `c2` pueden asignarse después del día en que se planificó el otro contenido.

Funciones:

- `(duracion-contenido ?d - contenido)`: Indica la duración del contenido `?d` en minutos.
- `(duracion-diaria ?dd - dia)`: Indica la duración total acumulada de los contenidos visualizados en el día `?dd`.

Acciones:

- `(ver (parameters (?c - contenido ?dia - dia)))`: Esta acción asegura que un contenido no puede ser visto si ya ha sido visto, y garantiza que se cumplan las relaciones de precedencia y paralelismo. Por último, comprueba que la duración total acumulada del día `?dia` no puede exceder el límite de 200 minutos al añadir el contenido `?c`. El efecto de esta acción marca el contenido `?c` como visto, lo asigna al día `?dia` y suma la duración del contenido `?c` al tiempo acumulado del día `?dia`.

3. Desarrollo de la Práctica

El desarrollo de esta práctica se ha llevado a cabo de forma incremental. Comenzamos implementando el modelo básico, estableciendo las bases para el resto de la práctica, como los predicados y los operadores. Una vez completado el nivel básico, pasamos a la extensión 1, aprovechando la implementación anterior y ajustándola para que cumpla el nuevo objetivo. Repetimos el mismo proceso con la extensión 2. Por último, para el resto de extensiones, la 3 y la 4, utilizamos la extensión 2 como base. Tras completar todas las extensiones, diseñamos un juego de prueba inicial para cada nivel y, posteriormente, implementamos un generador de juegos de prueba que crea problemas de forma aleatoria.

4. Juegos de Prueba

4.1. Extensión Básica

4.1.1. Juego de Prueba 1

Input:

```
(define (problem juego-de-prueba-basico-1)
    (:domain planner)
    (:objects C1 C2 C3 C4 - contenido)
    (:init
        (quiere-ver C1)
        (quiere-ver C2)
        (quiere-ver C3)
        (quiere-ver C4)
    )
    (:goal (and (forall (?c - contenido)
                        (imply (quiere-ver ?c) (ha-visto ?c))
                )
            )
    )
)
```

Output:

```
step    0: VER C4
        1: VER C3
        2: VER C2
        3: VER C1
```

En el primer juego de prueba, hemos considerado un caso muy sencillo en el que se desea visualizar todos los contenidos, sin que ninguno tenga un predecesor. Como se puede observar el planificador realiza la tarea correctamente.

4.1.2. Juego de Prueba 2

Input:

```
(define (problem juego-de-prueba-basico-2)
  (:domain planner)
  (:objects C0 C1 C2 C3 C4 C5 C6 C7 - contenido)
  (:init
    (predecesor C0 C2)
    (predecesor C2 C5)
    (predecesor C3 C4)
    (quiere-ver C1)
    (quiere-ver C5)
  )
```

```
  (:goal (and (forall (?c - contenido) (imply (quiere-ver ?c) (ha
     -visto ?c)))))
)
```

Output:

```
step    0: VER C0
        1: VER C2
        2: VER C5
        3: VER C1
```

Este juego de prueba se ha creado utilizando el generador. En este caso, se ha generado un problema en el que se desea visualizar los contenidos c1 y c5. El primer contenido no tiene predecesores, mientras que el contenido c5 tiene como predecesor al contenido c2, y este a su vez, tiene como predecesor al contenido c0. En el resultado, el planificador asigna primero c0 y c2 antes de c5. A continuación, se visualiza el contenido c1, cumpliendo así con el objetivo del programa.

4.2. Extensión 1

4.2.1. Juego de Prueba 1

Input:

```
(define (problem juego-de-prueba-ext1-1)
    (:domain plannerExt1)
    (:objects
        c1 c2 c3 c4 c5 - contenido
        d1 d2 d3 d4 d5 d6 d7 - dia
    )
    (:init
        (predecesor c1 c5)
        (predecesor c2 c5)
        (predecesor c3 c5)
        (predecesor c4 c5)
        (dia-anterior d1 d2)
        (dia-anterior d1 d3)
        (dia-anterior d1 d4)
        (dia-anterior d1 d5)
        (dia-anterior d1 d6)
        (dia-anterior d1 d7)
        (dia-anterior d2 d3)
        (dia-anterior d2 d4)
        (dia-anterior d2 d4)
        (dia-anterior d2 d5)
        (dia-anterior d2 d6)
        (dia-anterior d2 d7)
        (dia-anterior d3 d4)
        (dia-anterior d3 d5)
```

```
        (dia-anterior d3 d6)
        (dia-anterior d3 d7)
        (dia-anterior d4 d5)
        (dia-anterior d4 d6)
        (dia-anterior d4 d7)
        (dia-anterior d5 d6)
        (dia-anterior d5 d7)
        (dia-anterior d6 d7)
    )
    (:goal (and (ha-visto c5)))
)
```

Output:

```
step    0: VER C4 D1
        1: VER C3 D1
        2: VER C2 D1
        3: VER C1 D1
        4: VER C5 D7
```

En este caso ponemos a prueba que un contenido tenga más de un predecesor, como es el caso del contenido C5. Como podemos ver en la salida, se visualizan todos los predecesores de visualizarse C5.

4.2.2. Juego de Prueba 2

Input:

```
(define (problem juego-de-prueba-ext1-2)
  (:domain plannerExt1)
  (:objects
    C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 - contenido
    D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 - dia
  )
  (:init
    (predecesor C0 C5)
    (predecesor C1 C7)
    (predecesor C3 C7)
    (predecesor C6 C7)
    (predecesor C5 C8)
    (predecesor C7 C8)

    (dia-anterior D0 D1)
    (dia-anterior D0 D2)
    (dia-anterior D0 D3)
    (dia-anterior D0 D4)
    (dia-anterior D0 D5)
    (dia-anterior D0 D6)
    (dia-anterior D0 D7)
    (dia-anterior D0 D8)
```

```
    (dia-anterior D0 D9)
    (dia-anterior D1 D2)
    (dia-anterior D1 D3)
    (dia-anterior D1 D4)
    (dia-anterior D1 D5)
    (dia-anterior D1 D6)
    (dia-anterior D1 D7)
    (dia-anterior D1 D8)
    (dia-anterior D1 D9)
    (dia-anterior D2 D3)
    (dia-anterior D2 D4)
    (dia-anterior D2 D5)
    (dia-anterior D2 D6)
    (dia-anterior D2 D7)
    (dia-anterior D2 D8)
    (dia-anterior D2 D9)
    (dia-anterior D3 D4)
    (dia-anterior D3 D5)
    (dia-anterior D3 D6)
    (dia-anterior D3 D7)
    (dia-anterior D3 D8)
    (dia-anterior D3 D9)
    (dia-anterior D4 D5)
    (dia-anterior D4 D6)
    (dia-anterior D4 D7)
    (dia-anterior D4 D8)
    (dia-anterior D4 D9)
    (dia-anterior D5 D6)
    (dia-anterior D5 D7)
    (dia-anterior D5 D8)
    (dia-anterior D5 D9)
    (dia-anterior D6 D7)
    (dia-anterior D6 D8)
    (dia-anterior D6 D9)
    (dia-anterior D7 D8)
    (dia-anterior D7 D9)
    (dia-anterior D8 D9)
  )
  (:goal (and
    (ha-visto C8)
    (ha-visto C9)
    (ha-visto C6)
    )
  )
)
```

Output:

```
step    0: VER C1 D0
        1: VER C3 D0
        2: VER C6 D0
```

```
3: VER C0 D0
4: VER C7 D1
5: VER C5 D1
6: VER C8 D9
7: VER C9 D9
```

Este juego de prueba se ha creado utilizando el generador. El objetivo del programa es ver C6, C8 y C9. En este juego de prueba comprobamos contenidos precedentes encadenados. Para visualizar C8 y hace falta haber visto C5 y C7. El primer predecesor depende de C0 y el segundo de C1, C3, C6. En los resultados podemos ver que C8 se visualiza después de los predecesores. Por último, también se ha marcado como visto el contenido C9, que no tenia ninguna dependencia.

4.3. Extensión 2

4.3.1. Juego de Prueba 1

Input:

```
(define (problem juego-de-prueba-ext2-1)
    (:domain plannerExt2)
    (:objects
        c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 - contenido
        d1 d2 d3 d4 d5 d6 d7 - dia
    )
    (:init
        (predecesor c1 c2)
        (predecesor c2 c3)
        (predecesor c3 c4)
        (predecesor c4 c5)
        (predecesor c5 c9)
        (predecesor c9 c10)
        (paralelo c1 c6)
        (paralelo c4 c7)
        (paralelo c8 c10)

        (dia-siguiente d2 d1)
        (dia-siguiente d3 d2)
        (dia-siguiente d4 d3)
        (dia-siguiente d5 d4)
        (dia-siguiente d6 d5)
        (dia-siguiente d7 d6)
        (dia-anterior d1 d2)
        (dia-anterior d1 d3)
        (dia-anterior d1 d4)
        (dia-anterior d1 d5)
        (dia-anterior d1 d6)
        (dia-anterior d1 d7)
        (dia-anterior d2 d3)
```

```
        (dia-anterior d2 d4)
        (dia-anterior d2 d4)
        (dia-anterior d2 d5)
        (dia-anterior d2 d6)
        (dia-anterior d2 d7)
        (dia-anterior d3 d4)
        (dia-anterior d3 d5)
        (dia-anterior d3 d6)
        (dia-anterior d3 d7)
        (dia-anterior d4 d5)
        (dia-anterior d4 d6)
        (dia-anterior d4 d7)
        (dia-anterior d5 d6)
        (dia-anterior d5 d7)
        (dia-anterior d6 d7)
    )
    (:goal (and (ha-visto c1) (ha-visto c5)))
)
```

Output:

```
step    0: VER C6 D1
        1: VER C1 D1
        2: VER C2 D2
        3: VER C7 D4
        4: VER C3 D3
        5: VER C4 D4
        6: VER C5 D7
```

En este juego de pruebas combinamos de forma muy sencilla los predecesores y paralelos. En el resultado vamos que C1 y C6 se visualizan el mismo día porque son contenidos paralelos. Por otro lado, se siguen cumpliendo el orden de los predecesores como podemos de C3 a C2 y de C2 a C1.

4.3.2. Juego de Prueba 2

Input:

```
(define (problem juego-de-prueba-ext2-89)
  (:domain plannerExt2)
  (:objects
    C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 - contenido
    D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 - dia
  )
  (:init
    (predecesor C0 C6)
    (predecesor C1 C9)
    (predecesor C2 C9)
    (predecesor C3 C8)
```

```
(paralelo C2 C0)
(paralelo C2 C4)
(paralelo C3 C4)
(paralelo C3 C4)

(dia-siguiente D1 D0)
(dia-siguiente D2 D1)
(dia-siguiente D3 D2)
(dia-siguiente D4 D3)
(dia-siguiente D5 D4)
(dia-siguiente D6 D5)
(dia-siguiente D7 D6)
(dia-siguiente D8 D7)
(dia-siguiente D9 D8)

(dia-anterior D0 D1)
(dia-anterior D0 D2)
(dia-anterior D0 D3)
(dia-anterior D0 D4)
(dia-anterior D0 D5)
(dia-anterior D0 D6)
(dia-anterior D0 D7)
(dia-anterior D0 D8)
(dia-anterior D0 D9)
(dia-anterior D1 D2)
(dia-anterior D1 D3)
(dia-anterior D1 D4)
(dia-anterior D1 D5)
(dia-anterior D1 D6)
(dia-anterior D1 D7)
(dia-anterior D1 D8)
(dia-anterior D1 D9)
(dia-anterior D2 D3)
(dia-anterior D2 D4)
(dia-anterior D2 D5)
(dia-anterior D2 D6)
(dia-anterior D2 D7)
(dia-anterior D2 D8)
(dia-anterior D2 D9)
(dia-anterior D3 D4)
(dia-anterior D3 D5)
(dia-anterior D3 D6)
(dia-anterior D3 D7)
(dia-anterior D3 D8)
(dia-anterior D3 D9)
(dia-anterior D4 D5)
(dia-anterior D4 D6)
(dia-anterior D4 D7)
(dia-anterior D4 D8)
(dia-anterior D4 D9)
```

```
    (dia-anterior D5 D6)
    (dia-anterior D5 D7)
    (dia-anterior D5 D8)
    (dia-anterior D5 D9)
    (dia-anterior D6 D7)
    (dia-anterior D6 D8)
    (dia-anterior D6 D9)
    (dia-anterior D7 D8)
    (dia-anterior D7 D9)
    (dia-anterior D8 D9)
  )
  (:goal (and
    (ha-visto C0)
    (ha-visto C9)
    (ha-visto C2)
    (ha-visto C4)
    )
  )
)
```

Output:

```
step    0: VER C4 D0
        1: VER C0 D0
        2: VER C2 D0
        3: VER C1 D0
        4: VER C9 D9
```

Este juego de prueba se ha creado utilizando el generador. En este caso ponemos a prueba un nivel mayor de paralelismo. Por esta razón, podemos ver como se visualizan un gran número de contenido el dia D0.

4.4. Extensión 3

4.4.1. Juego de Prueba 1

Input:

```
(define (problem juego-de-prueba-ext3-1)
    (:domain plannerExt3)
    (:objects
        c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 -
           contenido
        d1 d2 d3 d4 d5 d6 d7 - dia
    )
    (:init
        (predecesor c1 c2)
        (predecesor c2 c3)
        (predecesor c3 c4)
        (predecesor c4 c5)
```

```
        (predecesor c5 c6)
        (predecesor c6 c7)

        (paralelo c1 c8)
        (paralelo c1 c9)
        (paralelo c1 c10)
        (paralelo c1 c11)
        (paralelo c7 c12)
        (paralelo c7 c13)
        (paralelo c7 c14)
        (paralelo c7 c15)

        (dia-siguiente d2 d1)
        (dia-siguiente d3 d2)
        (dia-siguiente d4 d3)
        (dia-siguiente d5 d4)
        (dia-siguiente d6 d5)
        (dia-siguiente d7 d6)

        (dia-anterior d1 d2)
        (dia-anterior d1 d3)
        (dia-anterior d1 d4)
        (dia-anterior d1 d5)
        (dia-anterior d1 d6)
        (dia-anterior d1 d7)
        (dia-anterior d2 d3)
        (dia-anterior d2 d4)
        (dia-anterior d2 d4)
        (dia-anterior d2 d5)
        (dia-anterior d2 d6)
        (dia-anterior d2 d7)
        (dia-anterior d3 d4)
        (dia-anterior d3 d5)
        (dia-anterior d3 d6)
        (dia-anterior d3 d7)
        (dia-anterior d4 d5)
        (dia-anterior d4 d6)
        (dia-anterior d4 d7)
        (dia-anterior d5 d6)
        (dia-anterior d5 d7)
        (dia-anterior d6 d7)

        (= (num-asignados d1) 0)
        (= (num-asignados d2) 0)
        (= (num-asignados d3) 0)
        (= (num-asignados d4) 0)
        (= (num-asignados d5) 0)
        (= (num-asignados d6) 0)
        (= (num-asignados d7) 0)
    )
```

```
    (:goal (ha-visto c7))
)
```

Output:

```
step    0: VER C11 D1
        1: VER C10 D1
        2: VER C15 D7
        3: VER C14 D7
        4: VER C13 D6
        5: VER C12 D6
        6: VER C9 D2
        7: VER C8 D2
        8: VER C1 D1
        9: VER C2 D2
       10: VER C3 D3
       11: VER C4 D4
       12: VER C5 D5
       13: VER C6 D6
       14: VER C7 D7
```

En este juego de prueba hemos aumentado el número de contenidos paralelos para poner a prueba el límite de 3 contenidos diarios, lo que provoca que el plan dure más días. Podemos observar que para cualquier día, el numero de contenidos del resultado no supera los 3. Por ejemplo, el contenido C1 es paralelo con 4 contenidos, en cambio en el resultado, en el día D1 se visualizan C1, C10 y C11.

4.4.2. Juego de Prueba 2

Input:

```
(define (problem juego-de-prueba-ext3-2)
  (:domain plannerExt3)
  (:objects
    C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 - contenido
    D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 - dia
  )
  (:init
    (predecesor C0 C6)
    (predecesor C2 C4)
    (predecesor C3 C9)

    (paralelo C4 C6)
    (paralelo C8 C4)
    (paralelo C2 C3)
    (paralelo C9 C8)
    (paralelo C8 C2)
    (paralelo C1 C2)

    (dia-siguiente D1 D0)
```

```
(dia-siguiente D2 D1)
(dia-siguiente D3 D2)
(dia-siguiente D4 D3)
(dia-siguiente D5 D4)
(dia-siguiente D6 D5)
(dia-siguiente D7 D6)
(dia-siguiente D8 D7)
(dia-siguiente D9 D8)

(dia-anterior D0 D1)
(dia-anterior D0 D2)
(dia-anterior D0 D3)
(dia-anterior D0 D4)
(dia-anterior D0 D5)
(dia-anterior D0 D6)
(dia-anterior D0 D7)
(dia-anterior D0 D8)
(dia-anterior D0 D9)
(dia-anterior D1 D2)
(dia-anterior D1 D3)
(dia-anterior D1 D4)
(dia-anterior D1 D5)
(dia-anterior D1 D6)
(dia-anterior D1 D7)
(dia-anterior D1 D8)
(dia-anterior D1 D9)
(dia-anterior D2 D3)
(dia-anterior D2 D4)
(dia-anterior D2 D5)
(dia-anterior D2 D6)
(dia-anterior D2 D7)
(dia-anterior D2 D8)
(dia-anterior D2 D9)
(dia-anterior D3 D4)
(dia-anterior D3 D5)
(dia-anterior D3 D6)
(dia-anterior D3 D7)
(dia-anterior D3 D8)
(dia-anterior D3 D9)
(dia-anterior D4 D5)
(dia-anterior D4 D6)
(dia-anterior D4 D7)
(dia-anterior D4 D8)
(dia-anterior D4 D9)
(dia-anterior D5 D6)
(dia-anterior D5 D7)
(dia-anterior D5 D8)
(dia-anterior D5 D9)
(dia-anterior D6 D7)
(dia-anterior D6 D8)
```

```
    (dia-anterior D6 D9)
    (dia-anterior D7 D8)
    (dia-anterior D7 D9)
    (dia-anterior D8 D9)

    (= (num-asignados D0) 0)
    (= (num-asignados D1) 0)
    (= (num-asignados D2) 0)
    (= (num-asignados D3) 0)
    (= (num-asignados D4) 0)
    (= (num-asignados D5) 0)
    (= (num-asignados D6) 0)
    (= (num-asignados D7) 0)
    (= (num-asignados D8) 0)
    (= (num-asignados D9) 0)
  )
  (:goal (and
    (ha-visto C2)
    (ha-visto C6)
    (ha-visto C9)
    (ha-visto C8)
    (ha-visto C1)
    )
  )
)
```

Output:

```
step    0: VER C0 D0
        1: VER C6 D9
        2: VER C3 D8
        3: VER C2 D7
        4: VER C4 D8
        5: VER C8 D8
        6: VER C9 D9
        7: VER C1 D7
```

En esta prueba, tenemos un caso más reducido. De igual manera que en el caso anterior, para cualquier día del dominio no se superan los 3 contenidos.

4.5. Extensión 4

4.5.1. Juego de Prueba 1

Input:

```
(define (problem juego-de-prueba-ext4-1)
    (:domain plannerExt4)
    (:objects
```

```
        c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 -
           contenido
        d1 d2 d3 d4 d5 d6 d7 - dia
    )
    (:init
        (predecesor c1 c2)
        (predecesor c2 c3)
        (predecesor c3 c4)
        (predecesor c4 c5)
        (predecesor c5 c6)
        (predecesor c6 c7)

        (paralelo c1 c8)
        (paralelo c1 c9)
        (paralelo c1 c10)
        (paralelo c1 c11)
        (paralelo c7 c12)
        (paralelo c7 c13)
        (paralelo c7 c14)
        (paralelo c7 c15)

        (dia-siguiente d2 d1)
        (dia-siguiente d3 d2)
        (dia-siguiente d4 d3)
        (dia-siguiente d5 d4)
        (dia-siguiente d6 d5)
        (dia-siguiente d7 d6)

        (dia-anterior d1 d2)
        (dia-anterior d1 d3)
        (dia-anterior d1 d4)
        (dia-anterior d1 d5)
        (dia-anterior d1 d6)
        (dia-anterior d1 d7)
        (dia-anterior d2 d3)
        (dia-anterior d2 d4)
        (dia-anterior d2 d4)
        (dia-anterior d2 d5)
        (dia-anterior d2 d6)
        (dia-anterior d2 d7)
        (dia-anterior d3 d4)
        (dia-anterior d3 d5)
        (dia-anterior d3 d6)
        (dia-anterior d3 d7)
        (dia-anterior d4 d5)
        (dia-anterior d4 d6)
        (dia-anterior d4 d7)
        (dia-anterior d5 d6)
        (dia-anterior d5 d7)
        (dia-anterior d6 d7)
```

```
        (= (duracion-contenido c1) 40)
        (= (duracion-contenido c2) 30)
        (= (duracion-contenido c3) 90)
        (= (duracion-contenido c4) 60)
        (= (duracion-contenido c5) 20)
        (= (duracion-contenido c6) 40)
        (= (duracion-contenido c7) 50)
        (= (duracion-contenido c8) 100)
        (= (duracion-contenido c9) 30)
        (= (duracion-contenido c10) 30)
        (= (duracion-contenido c11) 50)
        (= (duracion-contenido c12) 10)
        (= (duracion-contenido c13) 20)
        (= (duracion-contenido c14) 20)
        (= (duracion-contenido c15) 40)

        (= (duracion-diaria d1) 0)
        (= (duracion-diaria d2) 0)
        (= (duracion-diaria d3) 0)
        (= (duracion-diaria d4) 0)
        (= (duracion-diaria d5) 0)
        (= (duracion-diaria d6) 0)
        (= (duracion-diaria d7) 0)
    )
    (:goal (ha-visto c7))
)
```

Output:

```
step    0: VER C0 D0
        1: VER C6 D9
        2: VER C3 D8
        3: VER C2 D7
        4: VER C4 D8
        5: VER C8 D8
        6: VER C9 D9
        7: VER C1 D7
```

4.5.2. Juego de Prueba 2

Input:

```
(define (problem juego-de-prueba-ext4-2)
  (:domain plannerExt4)
  (:objects
    C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C12 - contenido
    D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 - dia
  )
  (:init
```

```
(predecesor C0 C12)
(predecesor C1 C8)
(predecesor C2 C11)
(predecesor C3 C4)
(predecesor C4 C7)
(predecesor C5 C6)
(predecesor C6 C10)
(predecesor C7 C12)
(paralelo C0 C7)
(paralelo C8 C4)
(paralelo C1 C5)
(paralelo C6 C1)
(paralelo C0 C1)

(dia-siguiente D1 D0)
(dia-siguiente D2 D1)
(dia-siguiente D3 D2)
(dia-siguiente D4 D3)
(dia-siguiente D5 D4)
(dia-siguiente D6 D5)
(dia-siguiente D7 D6)
(dia-siguiente D8 D7)
(dia-siguiente D9 D8)
(dia-siguiente D10 D9)
(dia-siguiente D11 D10)
(dia-siguiente D12 D11)

(dia-anterior D0 D1)
(dia-anterior D0 D2)
(dia-anterior D0 D3)
(dia-anterior D0 D4)
(dia-anterior D0 D5)
(dia-anterior D0 D6)
(dia-anterior D0 D7)
(dia-anterior D0 D8)
(dia-anterior D0 D9)
(dia-anterior D0 D10)
(dia-anterior D0 D11)
(dia-anterior D0 D12)
(dia-anterior D1 D2)
(dia-anterior D1 D3)
(dia-anterior D1 D4)
(dia-anterior D1 D5)
(dia-anterior D1 D6)
(dia-anterior D1 D7)
(dia-anterior D1 D8)
(dia-anterior D1 D9)
(dia-anterior D1 D10)
(dia-anterior D1 D11)
(dia-anterior D1 D12)
```

```
(dia-anterior D2 D3)
(dia-anterior D2 D4)
(dia-anterior D2 D5)
(dia-anterior D2 D6)
(dia-anterior D2 D7)
(dia-anterior D2 D8)
(dia-anterior D2 D9)
(dia-anterior D2 D10)
(dia-anterior D2 D11)
(dia-anterior D2 D12)
(dia-anterior D3 D4)
(dia-anterior D3 D5)
(dia-anterior D3 D6)
(dia-anterior D3 D7)
(dia-anterior D3 D8)
(dia-anterior D3 D9)
(dia-anterior D3 D10)
(dia-anterior D3 D11)
(dia-anterior D3 D12)
(dia-anterior D4 D5)
(dia-anterior D4 D6)
(dia-anterior D4 D7)
(dia-anterior D4 D8)
(dia-anterior D4 D9)
(dia-anterior D4 D10)
(dia-anterior D4 D11)
(dia-anterior D4 D12)
(dia-anterior D5 D6)
(dia-anterior D5 D7)
(dia-anterior D5 D8)
(dia-anterior D5 D9)
(dia-anterior D5 D10)
(dia-anterior D5 D11)
(dia-anterior D5 D12)
(dia-anterior D6 D7)
(dia-anterior D6 D8)
(dia-anterior D6 D9)
(dia-anterior D6 D10)
(dia-anterior D6 D11)
(dia-anterior D6 D12)
(dia-anterior D7 D8)
(dia-anterior D7 D9)
(dia-anterior D7 D10)
(dia-anterior D7 D11)
(dia-anterior D7 D12)
(dia-anterior D8 D9)
(dia-anterior D8 D10)
(dia-anterior D8 D11)
(dia-anterior D8 D12)
(dia-anterior D9 D10)
```

```
    (dia-anterior D9 D11)
    (dia-anterior D9 D12)
    (dia-anterior D10 D11)
    (dia-anterior D10 D12)
    (dia-anterior D11 D12)

    (= (duracion-contenido C0) 34)
    (= (duracion-contenido C1) 92)
    (= (duracion-contenido C2) 73)
    (= (duracion-contenido C3) 70)
    (= (duracion-contenido C4) 66)
    (= (duracion-contenido C5) 68)
    (= (duracion-contenido C6) 83)
    (= (duracion-contenido C7) 87)
    (= (duracion-contenido C8) 85)
    (= (duracion-contenido C9) 65)
    (= (duracion-contenido C10) 45)
    (= (duracion-contenido C11) 69)
    (= (duracion-contenido C12) 38)

    (= (duracion-diaria D0) 0)
    (= (duracion-diaria D1) 0)
    (= (duracion-diaria D2) 0)
    (= (duracion-diaria D3) 0)
    (= (duracion-diaria D4) 0)
    (= (duracion-diaria D5) 0)
    (= (duracion-diaria D6) 0)
    (= (duracion-diaria D7) 0)
    (= (duracion-diaria D8) 0)
    (= (duracion-diaria D9) 0)
    (= (duracion-diaria D10) 0)
    (= (duracion-diaria D11) 0)
    (= (duracion-diaria D12) 0)
  )
  (:goal (and
    (ha-visto C4)
    (ha-visto C11)
    (ha-visto C5)
    (ha-visto C7)
    (ha-visto C3)
    (ha-visto C6)
    (ha-visto C0)
    )
  )
)
```

Output:

```
step    0: VER C3 D0
        1: VER C2 D0
        2: VER C4 D1
```

```
3: VER C5 D1
4: VER C7 D3
5: VER C1 D2
6: VER C11 D12
7: VER C6 D3
8: VER C0 D2
```