Práctica 3: REDFLIX

Planificación

Inteligencia Artificial

Laia Ondoño: laia.ondono@estudiantat.upc.edu
Paula Gené: paula.gene@estudiantat.upc.edu
Anna Llanza: anna.llanza@estudiantat.upc.edu

QT 2020-2021 10/1/2021

Índice

1. Dominio

Definir y modelar correctamente el dominio es esencial para obtener una buena planificación. Para ello es necesario tener una representación correcta de los objetos o variables (*types*) que nos interesa representar y sus propiedades (*predicates*), además de las acciones (*actions*) con las que podemos cambiar el estado en el que nos encontramos.

1.1. Nivel básico

En el plan de visionado del nivel básico, todos los contenidos tienen 0 o 1 predecesores y ningún paralelo. Para conseguir que el planificador cumpla estas restricciones, hemos definido las siguientes variables, predicados y acciones:

Types:

- *contenido*: tipo que representa el contenido audiovisual de REDFLIX (películas o capítulos de series televisivas).
- *dia*: tipo que representa un día en el que el usuario puede ver contenidos.

Predicates:

- *(predecesor ?c1 - contenido ?c2 - contenido)*: indica que el contenido ?c1 es predecesor de ?c2 y, por lo tanto, ?c1 se debe ver en un día anterior al que se ha planificado para ver ?c2.
- *(visto ?c - contenido)*: indica si el contenido ?c ya ha sido visto por el usuario.
- (asignado ?c - contenido): indica si el contenido ?c ya ha sido asignado a algún día del plan.
- *(pendiente ?c - contenido)*: indica si el contenido ?c aún tiene que ser asignado a un día del plan y está pendiente de ver por parte del usuario.
- *(tiene_contenido ?d - dia)*: indica si el día ?d ya tiene contenido asignado.

Actions:

- *asignar_contenido (?c - contenido ?d - dia)*: esta acción busca asignar el contenido ?c al día ?d cumpliendo la restricción de ver el predecesor en un día anterior a ?d.

1.2. Extensión 1

La primera extensión amplia el número máximo de predecesores que un contenido puede tener hasta N, pero siguen sin haber contenidos paralelos. Con este cambio, el dominio definido queda modificado de la siguiente manera:

Types:

- No se ven modificados, ver apartado 1.1. Nivel básico.

Predicates:

- No se ven modificados, ver apartado 1.1. Nivel básico.

Actions:

- *asignar_contenido (?c - contenido ?d - dia)*: esta acción busca asignar el contenido ?c al día ?d cumpliendo la restricción de todos los predecesores en días anteriores a ?d, asegurándose de que esta restricción se cumpla para todos los elementos de la cadena de predecesores.

1.3. Extensión 2

En esta extensión se añade un nuevo elemento a la extensión 1: los contenidos paralelos (un contenido puede tener hasta M contenidos paralelos). El modelado del dominio en este caso queda de la siguiente manera:

Types:

- No se ven modificados, ver apartado 1.1. Nivel básico.

Predicates:

- Nivel básico + :
- *(paralelo ?c1 - contenido ?c2 - contenido)*: indica que el contenido ?c1 es paralelo a ?c2 y, por lo tanto, ambos contenidos se deben ver en el mismo día o en días seguidos (anterior o siguiente).
- *(asignacion ?c - contenido ?d - dia)*: indica el día ?d en que en contenido ?c ha sido asignado.
- *(dia_siguiente ?d1 - dia ?d2 - dia)*: indica que ?d1 es el día anterior a ?d2.

Actions:

- *asignar_contenido (?c - contenido ?d - dia)*: "1.2. Extensión 1" + también debe tener en cuenta que todos los paralelos de ?c cumplan la restricción y sean asignados al mismo día ?d o en días consecutivos a éste.

1.4. Extensión 3

La tercera extensión añade una nueva restricción: que el planificador controle que no se coloquen más de 3 contenidos al día. Teniendo esto en cuenta, hemos realizado los siguientes cambios:

Types:

- No se ven modificados, ver apartado 1.1. Nivel básico.

Predicates:

- No se ven modificados, ver apartado 1.3. Extensión 2.

Functions:

- *(numAsignaciones ?d - dia)*: número de contenidos que ya han sido asignados al día ?d.

Actions:

- *asignar_contenido (?c - contenido ?d - dia)*: Extensión 2 + controlar que no se asignen más de 3 contenidos en un mismo día.

1.5. Extensión 4

Esta última extensión también parte de la extensión 2. En este caso, aparece un nuevo concepto: el número de minutos de duración de un contenido. La restricción añadida es que en el plan generado no se superen los 200 minutos de contenido al día. Partiendo de esta nueva restricción, el modelado del dominio queda modificado de la siguiente manera:

Types:

- No se ven modificados, ver apartado 1.1. Nivel básico.

Predicates:

- No se ven modificados, ver apartado 1.3. Extensión 2.

Functions:

- *(duracion ?c - contenido)*: duración en minutos del contenido ?c.
- *(duracionDiaria ?d - dia)*: duración acumulada de los contenidos planificados para ver el día ?d.
- *(numDia ?d - dia)*: número identificador de un objeto día ?d.

Actions:

- *asignar_contenido (?c - contenido ?d - dia)*: Extensión 2 + controlar que no se superen los 200 minutos de contenido al día.

2. Problema

En todas las diferentes versiones del problema existen 2 tipos de objetos. El objeto contenido que representa los contenidos audiovisuales de la empresa REDFLIX y el objeto día que representa cada uno de los días en los que se le puede asignar contenidos para así poder crear un plan de visualización.

2.1. Nivel básico

La extensión básica solamente pueden existir contenidos con ningún o 1 predecesor. Por lo tanto, en el estado inicial, no hay ningún contenido con más de 1 predecesor, ni ningún contenido que sea paralelo a otro.

En el estado inicial, se indica los contenidos predecesores con el predicado predecesor, los que el usuario ya ha visto con el predicado visto y los que quiere ver, usando el predicado pendiente.

En estado final, el problema debe llegar a mostrar un plan de visualización para que el usuario vea los contenidos que tiene pendiente, es por eso que como objetivo solo necesitamos que estén los contenidos pendientes como asignados, usando el predicado asignado.

2.2. Extensión 1

En esta extensión pueden haber contenidos con más de 1 predecesor, pero sin ningún paralelo. Lo único que lo diferencia de la extensión básica, es que para la definición del estado inicial, cabe la posibilidad de tener predicados predecesor que generen una cadena de más de 2 contenidos seguidos. Tanto los objetos como el estado final, se expresa de la misma manera que en el anterior.

2.3. Extensión 2

Añadiendo en el estado inicial a la extensión anterior, el predicado dia_siguiente y paralelo, podremos crear un plan de visualización con contenidos que puedan ser predecesores y paralelos. Es necesario declarar en el estado inicial el predicado dia_siguiente para indicar el orden de los objetos día creados. El predicado paralelo sirve para indicar que contenido es paralelo con otro. Por lo que hace a los objetos y el estado inicial se mantienen como en la extensión básica.

2.4. Extensión 3

La extensión 3 solo permite que se asignen como máximo 3 contenidos diarios. Es por eso, que para definir el estado inicial es necesario usar el predicado numAsignaciones para cada día e inicializarlo a 0. No hay ningún cambio para los objetos ni para el estado final.

2.5. Extensión 4

La última extensión, no se ven afectado los objetos ni el estado final, pero en el estado inicial se requiere de los nuevos predicados duracion para inicializar la duración de cada uno de los objetos contenido, numDia para indicar el número del día y duracionDiaria para inicializar a 0 los minutos diarios vistos por el usuario en el plan que se genera. De esta manera, se podrá crear un plan de visualización sin sobrepasar los 200 minutos diarios.

3. Desarrollo de los modelos

Para poder llevar a cabo esta práctica, antes de todo nos hemos leído todo el enunciado y organizado según el tiempo del que disponíamos para la entrega. Seguidamente, instalamos todo lo necesario para poder ejecutar los ficheros PDDL y tener un entorno de desarrollo adecuado.

Para implementar todas las extensiones que pide la práctica, hemos empezado con el nivel básico terminando con la extensión 4. La metodología utilizada consiste en un desarrollo incremental basado en las diferentes extensiones. Es decir, cada vez que se implementa una extensión, para hacer la siguiente, utilizamos como base la más reciente y de esta añadimos o modificamos todo lo necesario, excepto para la última extensión que hemos usado como base la extensión 2.

Finalmente para comprobar que los modelos están bien desarrollados, hemos creado un fichero en Python que sirve para generar juegos de prueba para cada una de las extensiones disponibles.

4. Juegos de prueba

Para poder comprobar el correcto funcionamiento de nuestro dominio y las acciones definidas en este hemos creído necesario documentar como mínimo dos juegos de prueba para cada una de las extensiones implementadas. Además, hemos elaborado el programa *generador.py* en lenguaje Python, que permite crear diferentes problemas con parámetros aleatorios. En este apartado de la documentación se puede encontrar un juego de prueba elaborado manualmente y otro generado utilizando el fichero generador para cada extensión.

4.1. Juego de prueba 1

Para el primer juego de prueba se utiliza el dominio del nivel básico, y hemos elaborado manualmente el siguiente código para el problema.

ENTRADA

```
(define (problem problemaPlanificador)
 (:domain dominioPlanificador)

 (:objects
        c1 c2 c3 c4 c5 c6 - contenido
        dia1 dia2 dia3 dia4 dia5 dia6 dia7 - dia
 )

 (:init
        (predecesor c4 c6)
        (predecesor c2 c3)
        (predecesor c1 c5)
        (visto c2)
        (pendiente c3)
        (pendiente c5)
        (pendiente c6)
 )

 (:goal (and (asignado c3) (asignado c5) (asignado c6))
 )
)
```

SALIDA

step 0: ASIGNAR-CONTENIDO C4 DIA1
1: ASIGNAR-CONTENIDO C6 DIA2
2: ASIGNAR-CONTENIDO C3 DIA3
3: ASIGNAR-CONTENIDO C1 DIA4
4: ASIGNAR-CONTENIDO C5 DIA5

JUSTIFICACIÓN

En este primer nivel básico todos los contenidos tienen cero o un predecesor y ningún paralelo. En este juego de prueba se trabaja con 6 contenidos, 3 de los cuales son predecesores de otros 3 de la siguiente manera: c4 predecesor de c6, c2 predecesor de c3, y c1 predecesor de c5. Suponemos que el usuario quiere visualizar c3, c5 y c6, y que previamente ya ha visualizado c2, con lo que le queda pendiente visualizar c1 y c4 antes de poder ver c5 y c6, respectivamente. Es por esto, tal y como vemos en la salida, que primero se asigna c4 para el día 1, antes de c6 para el día 2, c3 el día 3 (ya que no es necesario añadir c2), luego c1 el día 4 y, por último, c5 el día 5. De esta manera, queda comprobado el funcionamiento correcto del nivel básico en cuanto a las restricciones de los predecesores en el caso de uno.

4.2. Juego de prueba 2

Para el segundo juego de prueba se utiliza el dominio del nivel básico, y se genera el problema con el fichero generador con parámetros aleatorios.

ENTRADA

```
(define (problem problemaPlanificador)
  (:domain dominioPlanificador)

  (:objects
      c1 c2 c3 c4 c5 c6 c7 c8 c9 - contenido
      dia1 dia2 dia3 dia4 dia5 dia6 dia7 dia8 dia9 dia10 dia11 - dia
  )

  (:init
      (predecesor c1 c2)
      (predecesor c3 c4)
      (visto c1)
      (pendiente c2)
      (pendiente c4)
```

(pendiente c5)
(pendiente c6)
(pendiente c7)
(pendiente c8)
(pendiente c9)
)

(:goal (and (asignado c2) (asignado c4) (asignado c5) (asignado c6) (asignado c7) (asignado c8) (asignado c9))
)
)

SALIDA

step 0: ASIGNAR-CONTENIDO C9 DIA1
1: ASIGNAR-CONTENIDO C8 DIA2
2: ASIGNAR-CONTENIDO C7 DIA3
3: ASIGNAR-CONTENIDO C6 DIA4
4: ASIGNAR-CONTENIDO C5 DIA5
5: ASIGNAR-CONTENIDO C3 DIA6
6: ASIGNAR-CONTENIDO C4 DIA7
7: ASIGNAR-CONTENIDO C2 DIA8

JUSTIFICACIÓN

En este segundo caso para el nivel básico, contamos con 9 contenidos, y podemos ver que dos de estos tienen predecesores, que son c2 y c4, con c1 y c3 como predecesores, y el resto sin predecesores, que son c5, c6, c7, c8 y c9. Por otro lado, se indica que el contenido c1 ya ha sido visto, De esta manera, la salida muestra que primero se asigna el día a los contenidos que no tienen predecesores y finalmente a c3 (predecesor de c4), c4 y c2.

4.3. Juego de prueba 3

Para el tercer juego de prueba se utiliza el dominio del nivel básico más la primera extensión, y hemos elaborado manualmente el siguiente código para el problema.

ENTRADA

(define (problem problemaPlanificador)
(:domain dominioPlanificador)

(:objects
c1 c2 c3 c4 c5 c6 c7 - contenido

```
        dia1 dia2 dia3 dia4 dia5 dia6 dia7 - dia
 )

 (:init
        (predecesor c4 c6)
        (predecesor c2 c3)
        (predecesor c1 c5)
        (predecesor c7 c1)
        (visto c2)
        (pendiente c3)
        (pendiente c5)
        (pendiente c6)
 )

 (:goal (and (asignado c3) (asignado c5) (asignado c6))
 )
)
```

SALIDA

```
step    0: ASIGNAR_CONTENIDO C7 DIA1
        1: ASIGNAR_CONTENIDO C1 DIA2
        2: ASIGNAR_CONTENIDO C5 DIA3
        3: ASIGNAR_CONTENIDO C4 DIA4
        4: ASIGNAR_CONTENIDO C6 DIA5
        5: ASIGNAR_CONTENIDO C3 DIA6
```

JUSTIFICACIÓN

En este caso con la primera extensión, todos los contenidos pueden tener de cero a N predecesores, pero ningún paralelo. En este juego de prueba se trabaja con 7 contenidos, uno de los cuales tiene un predecesor que al mismo tiempo tiene otro (c5, c1 y c7, respectivamente), y otros dos que tienen un predecesor cada uno. Hemos elegido este juego de prueba para comprobar que el programa entiende correctamente el orden que esto implica. El usuario ya ha visto c2, y tiene la intención de ver c3, c5 y c6, con lo que antes de c5 tendrá que ver c1, y antes de c1 c7, y por otro lado, antes de c6, tendrá que ver c4. Tal y como vemos en la salida, primero se asigna c7 para el día 1, antes de c1 para el día 2, y c5 para el día 3. Por otro lado, c4 el día 4, antes de c6 el día 5, y por último, de manera independiente, c3 el día 6. Con esto queda comprobado el funcionamiento correcto de la primera extensión en cuanto a las restricciones de más de un predecesor.

4.4. Juego de prueba 4

Para el cuarto juego de prueba se utiliza el dominio del nivel básico más la primera extensión, y se genera el problema con el fichero generador con parámetros aleatorios.

ENTRADA

```
(define (problem problemaPlanificador)
   (:domain dominioPlanificador)

   (:objects
         c1 c2 c3 c4 c5 c6 c7 c8 - contenido
         dia1 dia2 dia3 dia4 dia5 dia6 dia7 dia8 dia9 dia10 dia11 - dia
   )

   (:init
         (predecesor c1 c2)
         (predecesor c2 c3)
         (predecesor c4 c5)
         (predecesor c5 c6)
         (visto c1)
         (pendiente c3)
         (pendiente c6)
         (pendiente c7)
         (pendiente c8)
   )

   (:goal (and (asignado c3) (asignado c6) (asignado c7) (asignado c8))
   )
)
```

SALIDA

```
step    0: ASIGNAR_CONTENIDO C8 DIA1
        1: ASIGNAR_CONTENIDO C7 DIA2
        2: ASIGNAR_CONTENIDO C4 DIA3
        3: ASIGNAR_CONTENIDO C5 DIA4
        4: ASIGNAR_CONTENIDO C6 DIA5
        5: ASIGNAR_CONTENIDO C2 DIA6
        6: ASIGNAR_CONTENIDO C3 DIA7
```

JUSTIFICACIÓN

En este segundo caso para el nivel básico con la primera extensión, contamos con 8 contenidos, y podemos encontrar dos cadenas de tres contenidos donde uno es predecesor de otro, y al mismo tiempo este es predecesor de otro. Es decir, c1 predecesor de c2 y c2 predecesor de c3, y lo mismo con c4, c5 y c6, respectivamente. Se indica que c1 ya se ha visto, y que el objetivo es ver c3, c6, c7 y c8, teniendo estos dos últimos cero predecesores. La salida muestra que primero se asigna el día a los contenidos que no tienen predecesores y luego asigna correctamente la cadena de c4, c5 y c6 en días sucesivos, y a continuación c2 y c3.

4.5. Juego de prueba 5

Para el quinto juego de prueba se utiliza el dominio del nivel básico más la primera y segunda extensiones, y hemos elaborado manualmente el siguiente código para el problema.

ENTRADA

```
(define (problem problemaPlanificador)
 (:domain dominioPlanificador)

 (:objects
        c1 c2 c3 c4 c5 c6 c7 - contenido
        dia1 dia2 dia3 dia4 dia5 dia6 dia7 - dia
 )

 (:init
        (predecesor c4 c6)
        (predecesor c2 c3)
        (predecesor c3 c4)
        (predecesor c1 c5)
        (predecesor c7 c1)
        (paralelo c6 c7)
        (visto c2)
        (pendiente c3)
        (pendiente c5)
        (pendiente c6)
        (dia_siguiente dia1 dia2)
        (dia_siguiente dia2 dia3)
        (dia_siguiente dia3 dia4)
        (dia_siguiente dia4 dia5)
```

```
        (dia_siguiente dia5 dia6)
        (dia_siguiente dia6 dia7)
 )

 (:goal (and (asignado c3) (asignado c5) (asignado c6))
 )
)
```

SALIDA

```
step    0: ASIGNAR_CONTENIDO C3 DIA1
        1: ASIGNAR_CONTENIDO C4 DIA2
        2: ASIGNAR_CONTENIDO C6 DIA3
        3: ASIGNAR_CONTENIDO C7 DIA3
        4: ASIGNAR_CONTENIDO C1 DIA4
        5: ASIGNAR_CONTENIDO C5 DIA5
```

JUSTIFICACIÓN

En este caso con la primera y segunda extensiones, todos los contenidos pueden tener de cero a N predecesores, y de cero a M contenidos paralelos. En este juego de prueba se trabaja con 7 contenidos, uno de los cuales tiene un predecesor que al mismo tiempo tiene otro (c5, c1 y c7, respectivamente), y otro que tiene un predecesor, que al mismo tiempo tiene un predecesor, el cual tiene otro (c2, c3, c4 y c6). Además, los contenidos c6 y c7 son paralelos, con lo cual deben visualizarse el mismo día, o con un día de diferencia. Hemos elegido este juego de prueba para comprobar que el programa entiende correctamente el funcionamiento de los contenidos paralelos, y con este caso encontramos contenidos que tienen varios predecesores, con lo que se puede comprobar que el programa cumple correctamente con orden que esto implica. El usuario ya ha visto c2, y tiene la intención de ver c3, c5 y c6, con lo que antes de c5 tendrá que ver c1, y antes de c1 c7, y por otro lado, antes de c6, tendrá que ver c4 y c3. Tal y como vemos en la salida, primero se asigna c3 para el día 1, antes de c4 para el día 2, y c6 para el día 3. Por otro lado, c7 también el día 3 (por el hecho de ser paralelo con c6), antes de c1 el día 4, y c5 el día 5. Con esto queda comprobado el funcionamiento correcto de la primera y segunda extensiones en cuanto a las restricciones de más de un predecesor e introducción de contenidos paralelos.

4.6. Juego de prueba 6

Para el sexto juego de prueba se utiliza el dominio del nivel básico más la primera y segunda extensiones, y se genera el problema con el fichero generador con parámetros aleatorios.

ENTRADA

```
(define (problem problemaPlanificador)
   (:domain dominioPlanificador)

   (:objects
         c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 - contenido
         dia1 dia2 dia3 dia4 dia5 dia6 dia7 dia8 dia9 dia10 dia11 dia12 dia13 dia14 dia15 dia16
dia17 dia18 - dia
   )

   (:init
         (predecesor c1 c2)
         (predecesor c2 c3)
         (predecesor c4 c5)
         (predecesor c5 c6)
         (predecesor c7 c8)
         (predecesor c8 c9)
         (paralelo c3 c4)
         (paralelo c6 c7)
         (visto c1)
         (pendiente c3)
         (pendiente c6)
         (pendiente c9)
         (pendiente c10)
   )

   (:goal (and (asignado c3) (asignado c6) (asignado c9) (asignado c10))
   )
)
```

SALIDA

```
step    0: ASIGNAR_CONTENIDO C10 DIA1
        1: ASIGNAR_CONTENIDO C2 DIA1
        2: ASIGNAR_CONTENIDO C3 DIA2
        3: ASIGNAR_CONTENIDO C4 DIA2
        4: ASIGNAR_CONTENIDO C5 DIA3
        5: ASIGNAR_CONTENIDO C6 DIA4
        6: ASIGNAR_CONTENIDO C7 DIA4
        7: ASIGNAR_CONTENIDO C8 DIA5
        8: ASIGNAR_CONTENIDO C9 DIA6
```

JUSTIFICACIÓN

En este segundo caso para el nivel básico con la primera y segunda extensiones, contamos con 10 contenidos, y podemos encontrar tres cadenas de tres contenidos donde uno es predecesor de otro, y al mismo tiempo este es predecesor de otro. Es decir, c1 predecesor de c2 y c2 predecesor de c3, y lo mismo con c4, c5 y c6, y c7, c8 y c9, respectivamente. Además, c3 es paralelo de c4 y c6 es paralelo de c7. Se indica que c1 ya se ha visto, y que el objetivo es ver c3, c6, c9 y c10, teniendo este último cero predecesores. La salida muestra que primero se asigna el día al contenido que no tiene predecesores y luego asigna correctamente la cadena de c2 y c3 en días sucesivos (ya que ya ha visto c2), y a continuación las dos cadenas (c4 c5 c6, y c7 c8 c9) también en días sucesivos separadamente. Se cumplen las restricciones de los paralelos, ya que c3 y c4 se asignan al mismo día, y lo mismo pasa con c6 y c7.

4.7. Juego de prueba 7

Para el séptimo juego de prueba se utiliza el dominio del nivel básico más la primera, segunda y tercera extensiones, y hemos elaborado manualmente el siguiente código para el problema.

ENTRADA

```
(define (problem problemaPlanificador)
 (:domain dominioPlanificador)

 (:objects
        c1 c2 c3 c4 c5 c6 c7 c8 c9 - contenido
        dia1 dia2 dia3 dia4 dia5 dia6 dia7 - dia
 )

 (:init
        (predecesor c4 c6)
        (predecesor c2 c3)
        (predecesor c3 c4)
        (predecesor c1 c5)
        (predecesor c7 c1)
        (paralelo c6 c9)
        (paralelo c6 c7)
        (paralelo c8 c6)
        (visto c2)
```

```
        (pendiente c3)
        (pendiente c5)
        (pendiente c6)
        (pendiente c8)
        (pendiente c9)
        (dia_siguiente dia1 dia2)
        (dia_siguiente dia2 dia3)
        (dia_siguiente dia3 dia4)
        (dia_siguiente dia4 dia5)
        (dia_siguiente dia5 dia6)
        (dia_siguiente dia6 dia7)
        (= (numAsignaciones dia1) 0)
        (= (numAsignaciones dia2) 0)
        (= (numAsignaciones dia3) 0)
        (= (numAsignaciones dia4) 0)
        (= (numAsignaciones dia5) 0)
        (= (numAsignaciones dia6) 0)
        (= (numAsignaciones dia7) 0)
 )

 (:goal (and (asignado c3) (asignado c5) (asignado c6) (asignado c8) (asignado c9))
 )
)
```

SALIDA

```
step    0: ASIGNAR_CONTENIDO C3 DIA1
        1: ASIGNAR_CONTENIDO C4 DIA2
        2: ASIGNAR_CONTENIDO C6 DIA3
        3: ASIGNAR_CONTENIDO C7 DIA3
        4: ASIGNAR_CONTENIDO C8 DIA4
        5: ASIGNAR_CONTENIDO C9 DIA4
        6: ASIGNAR_CONTENIDO C1 DIA4
        7: ASIGNAR_CONTENIDO C5 DIA5
```

JUSTIFICACIÓN

En este caso con la primera, segunda y tercera extensiones, todos los contenidos pueden tener de cero a N predecesores, de cero a M contenidos paralelos, y además encontramos la restricción de que el usuario no pueda ver más de 3 contenidos en un mismo día. En este juego de prueba se trabaja con 9 contenidos. Al igual que en el juego de prueba 5, uno de estos contenidos tiene un predecesor que al mismo tiempo tiene otro (c5, c1 y c7,

respectivamente), y otro tiene un predecesor, que al mismo tiempo tiene un predecesor, el cual tiene otro (c2, c3, c4 y c6), y c8 y c9 no tienen relaciones de predecesores. Además, los contenidos c6 y c7, c6 y c8, y c6 y c9 son paralelos, con lo cual deben visualizarse el mismo día, o con un día de diferencia. Hemos elegido este juego de prueba porque en primer lugar, hemos comprobado que el programa intenta asignar los contenidos paralelos en un mismo día, así que al añadir más relaciones de este tipo podremos comprobar que no se asignan más de tres para un mismo día. El usuario ya ha visto c2, y tiene la intención de ver c3, c5, c6, c8 y c9 con lo que antes de c5 tendrá que ver c1, y antes de c1 c7, y por otro lado, antes de c6, tendrá que ver c4 y c3, y al mismo tiempo cumplir las restricciones de paralelismo y de máximo de contenidos. Tal y como vemos en la salida, primero se asigna c3 para el día 1, antes de c4 para el día 2, y c6 para el día 3. Por otro lado, c7 también el día 3 (por el hecho de ser paralelo con c7), antes de c1 el día 4, y c5 el día 5. Los contenidos c8 y c9 al ser paralelos con c6 podrían asignarse el día 3 como c6, pero para no violar la restricción del máximo de 3 contenidos diarios se asignan al día 4. Con esto queda comprobado el funcionamiento correcto de la primera, segunda y tercera extensiones en cuanto a las restricciones de más de un predecesor y contenidos paralelos, con el máximo de 3 contenidos diarios.

4.8. Juego de prueba 8

Para el octavo juego de prueba se utiliza el dominio del nivel básico más la primera, segunda y tercera extensiones, y se genera el problema con el fichero generador con parámetros aleatorios.

ENTRADA

```
(define (problem problemaPlanificador)
  (:domain dominioPlanificador)

  (:objects
        c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 - contenido
        dia1 dia2 dia3 dia4 dia5 dia6 dia7 dia8 dia9 dia10 dia11 dia12 dia13 dia14 dia15 - dia
  )

  (:init
        (predecesor c1 c2)
        (predecesor c2 c3)
        (predecesor c4 c5)
        (predecesor c5 c6)
        (paralelo c3 c4)
        (visto c1)
        (pendiente c3)
```

```
        (pendiente c6)
        (pendiente c7)
        (pendiente c8)
        (pendiente c9)
        (pendiente c10)
        (dia_siguiente dia1 dia2)
        (dia_siguiente dia2 dia3)
        (dia_siguiente dia3 dia4)
        (dia_siguiente dia4 dia5)
        (dia_siguiente dia5 dia6)
        (dia_siguiente dia6 dia7)
        (dia_siguiente dia7 dia8)
        (dia_siguiente dia8 dia9)
        (dia_siguiente dia9 dia10)
        (dia_siguiente dia10 dia11)
        (dia_siguiente dia11 dia12)
        (dia_siguiente dia12 dia13)
        (dia_siguiente dia13 dia14)
        (dia_siguiente dia14 dia15)
        (= (numAsignaciones dia1) 0)
        (= (numAsignaciones dia2) 0)
        (= (numAsignaciones dia3) 0)
        (= (numAsignaciones dia4) 0)
        (= (numAsignaciones dia5) 0)
        (= (numAsignaciones dia6) 0)
        (= (numAsignaciones dia7) 0)
        (= (numAsignaciones dia8) 0)
        (= (numAsignaciones dia9) 0)
        (= (numAsignaciones dia10) 0)
        (= (numAsignaciones dia11) 0)
        (= (numAsignaciones dia12) 0)
        (= (numAsignaciones dia13) 0)
        (= (numAsignaciones dia14) 0)
        (= (numAsignaciones dia15) 0)
    )

    (:goal (and (asignado c3) (asignado c6) (asignado c7) (asignado c8) (asignado c9) (asignado
c10))
    )
)
```

SALIDA

```
step    0: ASIGNAR_CONTENIDO C2 DIA1
        1: ASIGNAR_CONTENIDO C3 DIA2
        2: ASIGNAR_CONTENIDO C4 DIA2
        3: ASIGNAR_CONTENIDO C5 DIA3
        4: ASIGNAR_CONTENIDO C6 DIA4
        5: ASIGNAR_CONTENIDO C7 DIA4
        6: ASIGNAR_CONTENIDO C8 DIA4
        7: ASIGNAR_CONTENIDO C9 DIA5
        8: ASIGNAR_CONTENIDO C10 DIA5
```

JUSTIFICACIÓN

En este segundo caso para el nivel básico con la primera, segunda y tercera extensiones, contamos con 10 contenidos, y similar al juego de prueba 6, podemos encontrar dos cadenas de tres contenidos donde uno es predecesor de otro, y al mismo tiempo este es predecesor de otro. Es decir, c1 predecesor de c2 y c2 predecesor de c3, y lo mismo con c4, c5 y c6, respectivamente. Además, c3 es paralelo de c4. Se indica que c1 ya se ha visto, y que el objetivo es ver c3, c6, c7, c8, c9 y c10, teniendo estos cuatro últimos cero predecesores. La salida muestra que se asignan días consecutivos a los contenidos que tienen relaciones de predecesores (c2 c3, y c4 c5 c6) y c3 y c4 se asignan al mismo día por ser paralelos. Luego se asignan los contenidos independientes c7, c8, c9 y c10, que empieza asignándolos al día 4, pero cuando este llega a tener 3 contenidos asignados, los dos restantes se asignan para el día siguiente.

4.9. Juego de prueba 9

Para el noveno juego de prueba se utiliza el dominio del nivel básico más la primera, segunda y cuarta extensiones, y hemos elaborado manualmente el siguiente código para el problema.

ENTRADA

```
(define (problem problemaPlanificador)
 (:domain dominioPlanificador)

 (:objects
        c1 c2 c3 c4 c5 c6 c7 c8 c9 - contenido
        dia1 dia2 dia3 dia4 dia5 dia6 dia7 - dia
 )
```

```
(:init
        (= (duracion c1) 40)
        (= (duracion c2) 30)
        (= (duracion c3) 90)
        (= (duracion c4) 60)
        (= (duracion c5) 20)
        (= (duracion c6) 40)
        (= (duracion c7) 50)
        (= (duracion c8) 100)
        (= (duracion c9) 70)
        (predecesor c4 c6)
        (predecesor c2 c3)
        (predecesor c3 c4)
        (predecesor c1 c5)
        (predecesor c7 c1)
        (paralelo c6 c9)
        (paralelo c6 c7)
        (paralelo c8 c6)
        (visto c2)
        (pendiente c3)
        (pendiente c5)
        (pendiente c6)
        (pendiente c8)
        (pendiente c9)
        (dia_siguiente dia1 dia2)
        (dia_siguiente dia2 dia3)
        (dia_siguiente dia3 dia4)
        (dia_siguiente dia4 dia5)
        (dia_siguiente dia5 dia6)
        (dia_siguiente dia6 dia7)
        (= (numDia dia1) 1)
        (= (numDia dia2) 2)
        (= (numDia dia3) 3)
        (= (numDia dia4) 4)
        (= (numDia dia5) 5)
        (= (numDia dia6) 6)
        (= (numDia dia7) 7)
        (= (duracionDiaria dia1) 0)
        (= (duracionDiaria dia2) 0)
        (= (duracionDiaria dia3) 0)
```

```
        (= (duracionDiaria dia4) 0)
        (= (duracionDiaria dia5) 0)
        (= (duracionDiaria dia6) 0)
        (= (duracionDiaria dia7) 0)
 )

 (:goal (and (asignado c3) (asignado c5) (asignado c6) (asignado c8) (asignado c9))
 )
)
```

SALIDA

```
step    0: ASIGNAR_CONTENIDO C3 DIA1
        1: ASIGNAR_CONTENIDO C4 DIA2
        2: ASIGNAR_CONTENIDO C6 DIA3
        3: ASIGNAR_CONTENIDO C7 DIA3
        4: ASIGNAR_CONTENIDO C8 DIA4
        5: ASIGNAR_CONTENIDO C9 DIA4
        6: ASIGNAR_CONTENIDO C1 DIA5
        7: ASIGNAR_CONTENIDO C5 DIA6
```

JUSTIFICACIÓN

En este caso con la primera, segunda y cuarta extensiones, todos los contenidos pueden tener de cero a N predecesores, de cero a M contenidos paralelos, y además encontramos la restricción de que el usuario no pueda ver más de 200 minutos de contenido en un mismo día. Este juego de prueba es el mismo que el número 7, pero se asignan diferentes duraciones a los contenidos para poder comprobar que se cumple la restricción del máximo de minutos diarios. Recordamos que se trabaja con 9 contenidos, uno de los cuales tiene un predecesor que al mismo tiempo tiene otro (c5, c1 y c7, respectivamente), y otro tiene un predecesor, que al mismo tiempo tiene un predecesor, el cual tiene otro (c2, c3, c4 y c6), y c8 y c9 no tienen relaciones de predecesores. Además, los contenidos c6 y c7, c6 y c8, y c6 y c9 son paralelos, con lo cual deben visualizarse el mismo día, o con un día de diferencia. Hemos elegido este juego de prueba porque en primer lugar, hemos comprobado que el programa intenta asignar los contenidos paralelos en un mismo día, así que al añadir más relaciones de este tipo podremos comprobar que no se asignan más de 200 minutos para un mismo día. El usuario ya ha visto c2, y tiene la intención de ver c3, c5, c6, c8 y c9 con lo que antes de c5 tendrá que ver c1, y antes de c1 c7, y por otro lado, antes de c6, tendrá que ver c4 y c3, y al mismo tiempo cumplir las restricciones de paralelismo y de máximo de contenidos. Tal y como vemos en la salida, primero se asigna c3 para el día 1, antes de c4 para el día 2, y c6 para el día 3. Por otro lado, c7 también el día 3 (por el hecho de ser paralelo con c7), antes de c1 el día 5 (el cual podría asignarse al día 4 por tener predecesor, pero se superaría el máximo de minutos y se asigna al 5) y c5 el día 6. Los

contenidos c8 y c9 al ser paralelos con c6 podrían asignarse el día 3 como c6, pero para no violar la restricción del máximo de 200 minutos diarios se asignan al día 4. Con esto queda comprobado el funcionamiento correcto de la primera, segunda y cuarta extensiones en cuanto a las restricciones de más de un predecesor y contenidos paralelos, con el máximo de 200 minutos de contenido diarios.

4.10. Juego de prueba 10

Para el décimo juego de prueba se utiliza el dominio del nivel básico más la primera, segunda y cuarta extensiones, y se genera el problema con el fichero generador con parámetros aleatorios.

ENTRADA

```
(define (problem problemaPlanificador)
  (:domain dominioPlanificador)

  (:objects
      c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 - contenido
      dia1 dia2 dia3 dia4 dia5 dia6 dia7 dia8 dia9 dia10 dia11 dia12 dia13 - dia
  )

  (:init
      (= (duracion c1) 122)
      (= (duracion c2) 127)
      (= (duracion c3) 57)
      (= (duracion c4) 115)
      (= (duracion c5) 80)
      (= (duracion c6) 79)
      (= (duracion c7) 103)
      (= (duracion c8) 39)
      (= (duracion c9) 41)
      (= (duracion c10) 54)
      (= (duracion c11) 30)
      (= (duracion c12) 91)
      (predecesor c1 c2)
      (predecesor c2 c3)
      (predecesor c4 c5)
      (predecesor c5 c6)
      (paralelo c3 c4)
      (visto c1)
      (pendiente c3)
```

(pendiente c6)
(pendiente c7)
(pendiente c8)
(pendiente c9)
(pendiente c10)
(pendiente c11)
(pendiente c12)
(dia_siguiente dia1 dia2)
(dia_siguiente dia2 dia3)
(dia_siguiente dia3 dia4)
(dia_siguiente dia4 dia5)
(dia_siguiente dia5 dia6)
(dia_siguiente dia6 dia7)
(dia_siguiente dia7 dia8)
(dia_siguiente dia8 dia9)
(dia_siguiente dia9 dia10)
(dia_siguiente dia10 dia11)
(dia_siguiente dia11 dia12)
(dia_siguiente dia12 dia13)
(= (numDia dia1) 1)
(= (numDia dia2) 2)
(= (numDia dia3) 3)
(= (numDia dia4) 4)
(= (numDia dia5) 5)
(= (numDia dia6) 6)
(= (numDia dia7) 7)
(= (numDia dia8) 8)
(= (numDia dia9) 9)
(= (numDia dia10) 10)
(= (numDia dia11) 11)
(= (numDia dia12) 12)
(= (numDia dia13) 13)
(= (duracionDiaria dia1) 0)
(= (duracionDiaria dia2) 0)
(= (duracionDiaria dia3) 0)
(= (duracionDiaria dia4) 0)
(= (duracionDiaria dia5) 0)
(= (duracionDiaria dia6) 0)
(= (duracionDiaria dia7) 0)
(= (duracionDiaria dia8) 0)
(= (duracionDiaria dia9) 0)

```
        (= (duracionDiaria dia10) 0)
        (= (duracionDiaria dia11) 0)
        (= (duracionDiaria dia12) 0)
        (= (duracionDiaria dia13) 0)
    )

    (:goal (and (asignado c3) (asignado c6) (asignado c7) (asignado c8) (asignado c9) (asignado
c10) (asignado c11) (asignado c12))
    )
)
```

SALIDA

```
step    0: ASIGNAR_CONTENIDO C2 DIA1
        1: ASIGNAR_CONTENIDO C3 DIA2
        2: ASIGNAR_CONTENIDO C4 DIA2
        3: ASIGNAR_CONTENIDO C5 DIA3
        4: ASIGNAR_CONTENIDO C6 DIA4
        5: ASIGNAR_CONTENIDO C7 DIA4
        6: ASIGNAR_CONTENIDO C8 DIA5
        7: ASIGNAR_CONTENIDO C9 DIA5
        8: ASIGNAR_CONTENIDO C10 DIA5
        9: ASIGNAR_CONTENIDO C11 DIA5
        10: ASIGNAR_CONTENIDO C12 DIA6
```

JUSTIFICACIÓN

En este segundo caso para el nivel básico con la primera, segunda y cuarta extensiones, contamos con 12 contenidos, y, al igual que en el juego de prueba 8, podemos encontrar dos cadenas de tres contenidos donde uno es predecesor de otro, y al mismo tiempo este es predecesor de otro. Es decir, c1 predecesor de c2 y c2 predecesor de c3, y lo mismo con c4, c5 y c6, respectivamente. Además, c3 es paralelo de c4. Se asignan duraciones a todos los contenidos, se indica que c1 ya se ha visto, y que el objetivo es ver c3, c6, c7, c8, c9, c10, c11 y c12, teniendo estos seis últimos cero predecesores.

La salida muestra que se asignan días consecutivos a los contenidos que tienen relaciones de predecesores (c2 c3, y c4 c5 c6) y c3 y c4 se asignan al mismo día por ser paralelos. Luego se asignan los contenidos independientes c7, c8, c9, c10, c11 y c12 que empieza asignándoles al día 4, pero cuando la suma de la duración de los contenidos asignados a este llega a 200 minutos, se pasa a asignar los contenidos c8, c19, c10 y c11 al día 5, hasta que pasa lo mismo y el contenido c12 tiene que asignarse al día 6.