Documentación de la práctica de Planificación

1er Cuatrimestre - curso 2016/2017

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Mario Fernández Villalba

Juan Miguel de Haro Ruiz

Carlos Roldán Montaner

ÍNDICE

1. Descripción del modelado del dominio

1.1 Nivel básico

En esta versión encontramos el tipo Content, que indica que un objeto es un contenido audiovisual, es decir, una película o capítulo de serie que aparece en el catálogo de REDFLIX. Este tipo también aparecerá en el resto de versiones, así que para evitar repeticiones, en cada extensión sólo mencionaremos los elementos que son nuevos y no aparecen en anteriores versiones.

De predicados, podemos encontrar los siguientes:

- (*predecessor* ?C1 - Content ?C2 - Content): Indica que el contenido C1 se tiene que visualizar antes que C2 ya que C1 es predecesor de C2.
- (*asignedContent* ?C - Content): Este predicado nos dice que el contenido C está asignado, es decir, ya se ha incluido en la plan de visualizado.
- (*asignedPredecessor* ?C - Content)**:** Como en el nivel básico cada contenido solo puede tener 0 o 1 predecesores, este predicado indica que el posible predecesor de C ya se ha incluido en el plan.
- (*desiredContent* ?C - Content): El contenido C es deseado, esto quiere decir que el usuario quiere ver ese contenido.

Los operadores son los siguientes:

- *makeDesired*: Puede pasar que un contenido deseado tenga un predecesor que no lo sea. En este caso, el operador permite convertir en deseados a todos los predecesores de un contenido que no lo sean. De esta manera, el usuario verá siempre todos los predecesores de cualquier contenido que sea deseado (en caso de que no los haya visto).

- *asignContentToDay*: Este operador es el encargado de incluir un contenido en el plan de visualización, en caso de que cumpla las condiciones establecidas en el nivel básico.

Aunque tengamos un operador que se llame *asignContentToDay*, en realidad no existe el concepto de día en el dominio, ya que no hay restricciones obliguen a visualizar todos los contenidos en más de un día. Por lo que en esta versión suponemos que todo se visualiza en el mismo día, en el orden que establece el plan. El funcionamiento se basa en el algoritmo de ordenación topológica de un grafo dirigido acíclico. Cuando se asigna un contenido C al plan (porque es deseado y su posible predecesor ya se ha asignado), se hace un recorrido por todos los contenidos de los cuales el contenido C es predecesor, y se indica que ya se ha asignado su único predecesor (con el predicado *asignedPredecessor*), por lo tanto todos esos contenidos se podrán asignar.

1.2 Extensión 1

En esta extensión aparece por primera vez el tipo Day, que representa un día en el cual se pueden visualizar contenidos.

Además, para simplificar el código y la complejidad de las precondiciones, decidimos usar fluentes. En esta extensión podemos encontrar los siguientes:

- (*predecessorsToAsign* ?content - Content): Esta función sustituye al predicado *assignedContent* del nivel básico. Su significado es el mismo, pero como ahora un contenido puede tener más de un predecesor, este fluente nos indica cuántos predecesores de ?content quedan aún por asignar al plan.

- (*numDay* ?D - Day): Un objeto de tipo día no proporciona ningún tipo de información. Hay que saber cuál va antes que otro, o después. Para eso decidimos usar un fluente para identificar cada día con un número que será diferente para cada uno. De esta manera el planificador puede saber de manera sencilla en qué orden están los días, y cuál va antes que otro, simplemente comparando el número que contiene la función *numDay*.

- (*maxDay* ?C - Content): En esta extensión todos los predecesores de un contenido se tienen que ver en días anteriores. Este fluente nos indica el día a partir del cual se puede asignar el contenido ?C, ya que si se asignara antes, no se cumpliría la condición. Por ejemplo, si *maxDay* es 5, el contenido ?C solo se podrá asignar en el día 6 o posterior.

El único operador que cambia es el *asignContentToDay*, el cual ya incluye como parámetro el día en el cual se asignará un contenido. Su función es la misma, pero ahora incluye las nuevas condiciones. Para saber si todos los predecesores se han asignado, solo hay que mirar que el fluente *predecessorsToAsign* sea 0, y para asegurar que la condición de los días se cumple, hay que comparar el número del día en el cual se quiere asignar el contenido con el fluente *maxDay*.

El efecto del operador hará lo mismo que en el nivel básico, pero como un contenido puede tener más de un predecesor, se decrementará en 1 el número de predecesores por asignar de todos los contenidos que se tengan que ver después del contenido asignado. De esta manera, cuando se hayan asignado todos los predecesores de un contenido, el fluente *predecessorsToAssign* será 0. A parte de esto, se puede observar otro forall en el efecto del operador. Éste es el que actualiza el fluente *maxDay*. Si el contenido C se ha asignado en un día posterior al indicado por *maxDay* para algún contenido que se tiene que visualizar después de C, se asigna a esta función el día de asignación de C. De esta manera, una vez se han incluido en el plan todos los predecesores de un contenido, *maxDay* indica el día máximo en el cual se ha asignado alguno de sus predecesores.

La actualización de *maxDay* se podría realizar en el mismo forall que decrementa *predecessorsToAsign*, pero el fast forward daba un error al poner más de un when en un mismo forall, por eso decidimos separarlos.

1.3 Extensión 2

La extensión 2 incluye los contenidos paralelos, y podemos encontrar un nuevo predicado:

- (*parallel* ?C1 - Content ?C2 -Content): Indica que los contenidos C1 y C2 son paralelos, por lo tanto se tienen que ver en un intervalo de 2 días y en cualquier orden. La relación es bidireccional, es decir, si existe el predicado (parallel c1 c2) también existirá (parallel c2 c1).

En la parte de funciones también aparece un nuevo fluente:

- (*dayContent* ?C - Content): Indica el día en el cual se ha asignado el contenido ?C.

Para poder incluir a los contenidos paralelos, decidimos usar un forall en la precondición. Éste obliga a que todos los contenidos que sean paralelos a C (el que se quiere incluir en el plan) y que estén asignados, lo estén en el día anterior, el siguiente o el mismo día al cual se quiere asignar C. La comprobación para saber si dos contenidos son paralelos se hace en un solo sentido, por eso se requiere que cada predicado esté duplicado. Con este forall nos aseguramos que se cumple la condición que indica el enunciado, ya que todos los contenidos que sean paralelos se verán en un intervalo de dos días seguidos. Para saber el día en el cual se ha asignado un contenido que es paralelo a C, se utiliza el fluente *dayContent*, el cual lo tendremos que asignar en la parte effect del operador. Si asignamos un contenido, hay que actualizar *dayContent* con el número del día (indicado por *numDay*) en el cual se asigna el contenido.

1.4 Extensión 3

Esta versión nos añade una restricción nueva, que no permite asignar más de 3 contenidos a un día, por eso podemos encontrar un nuevo fluente:

- (*numContentDay* ?D - Day): Este fluente nos indica cuántos contenidos se han asignado a un día.

Los cambios realizados en esta extensión son pocas. En la precondición de *asignContentToDay* se comprueba que el número de contenidos asignados al día sea menor que 3. En la parte effect del operador simplemente tendremos que hacer un increase del fluente *numContentDay* para indicar que se ha asignado un contenido a un día concreto.

1.5 Extensión 4

La extensión 4 es muy parecida a la 3. Como fluentes nuevos tenemos los siguientes (*numContentDay* no se incluye en esta versión):

- (*minDay* ?D - Day): Indica los minutos de visionado para un día concreto.
- (*minContent* ?C - Content): Indica los minutos que dura un contenido.

En la precondición de *asignContentToDay* habrá que comprobar que si asignamos el contenido a un día, los minutos de visionado para ese día no superen los 200 minutos. En el effect se actualiza minDay, sumándole los minutos que dura el contenido asignado (indicado por *minContent*). De esta manera se mantiene en todo momento cuántos minutos se visualizan cada día, y se puede hacer una comparación numérica simple.

2. Descripción del modelado de problemas

Para el modelado de los problemas disponemos de una serie de objetos de dos tipos: Contenido y Día. En los problemas generados por el generador de problemas aleatorios crearemos el mismo número de instancias de tipo Contenido y de tipo Día, de esta manera garantizaremos que el problema tenga días suficientes para poder obtener una solución. Esto es cierto para todas las extensiones, pero no para el nivel básico, en el que no hay días.

La inicialización varía según la extensión en la que nos encontramos.

2.1 Nivel Básico

En la inicialización, establecemos qué contenidos son predecesores de otros (mediante el predicado *predecessor*) y cuáles son los que el usuario desea ver (mediante el predicado *desiredContent*). También determinamos qué contenidos ya han sido vistos por el usuario (mediante *asignedContent*). Asimismo, inicializamos *assignedPredecessor* para indicar que los predecesores de un contenido están asignados. Si un contenido no tiene predecesores también debemos inicializarlo ya que si no, no se cumpliría la precondición del operador *asignContentToDay*.

2.2 Extensión 1

Además de lo anterior, establecemos para cada contenido cuántos predecesores faltan por asignar mediante la función *predecessorsToAsign*. En caso de que un contenido no tenga predecesores, aún así debemos inicializar *predecessorsToAsign* a 0 ya que es una condición del operador *makeDesired*. Asignamos a cada día su número correspondiente (función *numDay*) empezando en 1 y siendo los días consecutivos. Asimismo, inicializamos *maxDay* a 0 para que el operador pueda ejecutarse para los contenidos que no tienen predecesores, ya que así *maxDay* siempre será menor que *numDay*.

2.3 Extensión 2

Además de lo anterior, determinamos qué contenidos son paralelos a otros (mediante el predicado *parallel*).

2.4 Extensión 3

Además de lo anterior, para los problemas de la Extensión 3, inicializamos *numContentDay* a 0.

2.5 Extensión 4

Además de lo anterior, para los problemas de la Extensión 4, inicializamos los minutos ya vistos ese día a 0 (función *minDay*) y asignamos a cada contenido su tiempo de duración (función *minContent*). No obstante, no hay *numContentDay* ya que las extensiones 3 y 4 son exclusivas entre sí.

De ésta manera, como estado inicial tenemos una serie de días y de contenidos, y sabemos cuáles son predecesores y paralelos a cuáles. Algunos de estos contenidos son deseados por el usuario. Para cada contenido sabemos cuántos de sus predecesores faltan por asignar. En el caso de la extensión 4 tenemos también el tiempo de visualización de cada día inicializado a cero y el tiempo de visualización de cada contenido.

Como estado final, disponemos de todos los contenidos que el usuario desea ver, más todos los que sean necesarios para poder verlos, asignados al mínimo número de días respetando las restricciones de cada Extensión.

3. Descripción del desarrollo del sistema

Nos encontramos que esta práctica nos plantea crear una versión base y ir añadiendo extensiones a esta para mejorar su funcionamiento. Debido a esto hemos decidido seguir un desarrollo incremental, en el cual solo implementamos extensiones una vez finalizadas y comprobado el correcto funcionamiento de las anteriores.

4. Descripción de los juegos de prueba

Debido a que el desarrollo del sistema ha sido incremental, hemos ido diseñando juegos de prueba en cada fase de este para ir verificando el correcto funcionamiento de cada extensión. A continuación explicamos los juegos de prueba ejecutados en cada versión. Para ver el proceso de resolución de cada juego de prueba, consultar los archivos .txt correspondientes (están localizados junto a los juegos de prueba). Como todos los resultados han sido satisfactorios, no comentaremos nada más al respecto.

4.1 Juegos de prueba del nivel básico

Al ser esta la primera versión del dominio habrá que testear el funcionamiento correcto de los operadores *asignContentToDay* y *makeDesired*.

Juego de prueba 1: Testeo del operador *asignContentToDay*

Objetivo	Testear el correcto funcionamiento del operador *asignContentToDay*.
Descripción de la entrada	- Catálogo de 4 contenidos (*c1- c4)*. - Sin contenidos predecesores. - *c1* es un contenido ya visualizado. - Todos los contenidos no visualizados son deseados.
Descripción de la salida	El sistema ha de ser capaz de crear un plan de visualización ya que la única restricción del sistema es que si un contenido se visualiza, también han de verse sus predecesores, y esto no es problema al no haber contenidos predecesores. Además, en este plan no se incluirá a *c1*.

Juego de prueba 2: Testeo del operador *makeDesired*

Objetivo	Testear el correcto funcionamiento del operador *makeDesired*.
Descripción de la entrada	- Catálogo de 2 contenidos (*c1, c2)*. - *c1* es predecesor de *c2*. - Sin contenidos ya visualizados. - El contenido *c2* es deseado.
Descripción de la salida	El sistema ha de ser capaz de crear un plan de visualización y además ha de incluir en este a *c1* aunque no sea deseado, ya que este contenido es necesario para ver *c2*.

4.2 Juego de prueba de la Extensión 1

La extensión 1 añade la característica de que cada contenido puede tener más de un predecesor, así que esto es lo que testearemos en el juego de prueba.

Juego de prueba 3: Testeo de más de un predecesor por contenido

Objetivo	Testear el correcto funcionamiento del planificador con contenidos con más de un predecesor.
Descripción de la entrada	- Catálogo de 4 contenidos (*c1, c2, c3, c4)*. - 4 días disponibles. - *c1* y *c2* son predecesores de *c3, c3* es predecesor de *c4*. - Sin contenidos ya visualizados. - El contenido *c4* es deseado.
Descripción de la salida	El sistema ha de ser capaz de crear un plan de visualización y además ha de incluir en este todos los contenidos, ya que *c4* depende de *c3* y este depende a su vez de *c1* y *c2*.

4.3 Juegos de prueba de la Extensión 2

La extensión 2 añade la característica de que cada contenido puede tener contenidos paralelos, así que esto es lo que testearemos en el juego de prueba 4.

Juego de prueba 4: Testeo de los contenidos paralelos

Objetivo	Testear el correcto funcionamiento del planificador con contenidos paralelos.
Descripción de la entrada	- Catálogo de 3 contenidos (*c1, c2, c3)*. - 3 días disponibles. - Sin contenidos predecesores. - *c1* y *c2* son contenidos paralelos de *c3*. - Sin contenidos ya visualizados. - Todos los contenidos son deseados.
Descripción de la salida	El sistema ha de ser capaz de crear un plan de visualización y además *c1* y *c2* han de verse en el día anterior, posterior o en el mismo día que *c3* al ser paralelos a este.

Juego de prueba 5: Testeo de juego de prueba aleatorio

Para probar nuestro sistema con un problema complejo (uno que incluya predecesores, paralelos…) hemos decidido crear un juego de pruebas con el generador aleatorio de problemas. Este se encuentra en la carpeta Extension 2. Destacamos de este de juego de pruebas que hay contenidos deseados con predecesores pero estos no son asignados en la planificación, ya que los hemos visto previamente.

4.4 Juego de prueba de la Extensión 3

La extensión 3 añade a la extensión 2 la restricción de que solo se puedan ver cómo máximo 3 contenidos por día, así que esto es lo que testearemos en el juego de prueba.

Juego de prueba 6: Testeo de la restricción de contenidos por día

Objetivo	Testear el correcto funcionamiento del planificador con la restricción de 3 contenidos por día como máximo.
Descripción de la entrada	- Catálogo de 6 contenidos (*c1 - c6)*. - 3 días disponibles. - Sin contenidos predecesores. - Sin contenidos paralelos. - Sin contenidos ya visualizados. - Todos los contenidos son deseados.
Descripción de la salida	El sistema ha de ser capaz de crear un plan de visualización y además en cada día se han de ver cómo máximo 3 contenidos.

4.5 Juego de prueba de la Extensión 4

La extensión 4 añade a la extensión 2 la restricción de que cada día la suma de los tiempos de los contenidos asignados a éste sea menor o igual que 200 minutos, así que esto es lo que testearemos en el juego de prueba.

Juego de prueba 7: Testeo de la restricción de tiempo por día

Objetivo	Testear el correcto funcionamiento del planificador con la restricción de tiempo por día.
Descripción de la entrada	- Catálogo de 3 contenidos (*c1, c2 y c3)*. - 2 días disponibles. - Sin contenidos predecesores. - Sin contenidos paralelos. - *c1* tiene una duración de 200 minutos, *c2* y *c3* de 100 minutos. - Sin contenidos ya visualizados. - Todos los contenidos son deseados.
Descripción de la salida	El sistema ha de ser capaz de crear un plan de visualización y además en un día se ha de ver exclusivamente *c1* y en otro día se han de ver *c2* y *c3*.