Inteligencia Artificial

Práctica de planificación

Junio 2020

Aleix Boné Ribó
Alex Herrero Pons
Moisés Balcells

Índice

1. El problema

2. Modelado del dominio

2.1. Nivel básico

2.1.1. Variables

Las variables que usamos son: `dia`, `ejercicio` y `nivel`. Que como indica su nombre representan los días, los ejercicios disponibles y los niveles de dificultad respectivamente.

2.1.2. Predicados

Para modelar el problema hemos usado los predicados que se describen a continuación:

1. Predicados para modelar el problema (derivados directamente del enunciado):
   - `(preparador ?x - ejercicio ?y - ejercicio)`: ejercicio x es preparador de y.
   - `(objetivo ?x - ejercicio ?n - nivel)`: objetivo marcado por el usuario.
2. Predicados para establecer un orden entre variables:
   - `(prev ?x - dia ?y - dia)`: orden de dias.
   - `(next-nivel ?x - nivel ?y - nivel)`: orden de niveles.
3. Predicados auxiliares:
   - `(last ?dia - dia ?ej - ejercicio)`: Ultimo ejercicio realizado el día `?d`
   - `(reached ?ej - ejercicio ?n - nivel)`: Nivel alcanzado para el ejercicio
   - `(realiza ?x - ejercicio ?n - nivel ?d - dia)`: Se realiza el ejercicio el día `?d` con nivel `?n`
   - `(hecho ?e - ejercicio ?d - dia)`: Igual que realiza pero sin la información del nivel.

2.1.3. Acciones

Para el nivel básico solo usamos una acción la cual llamamos `realizar-ejercicio`, ya que con las condiciones del problema básico se podía modelar todas las condiciones en la precondicion de este.

2.2. Extensión 1

Para la primera extensión solamente hemos añadido el predicado `(preparado ?e - ejercicio ?d - dia)` que nos indica si en el día `?d` se han hecho todos los ejercicios preparadores del ejercicio `?e`. Para computar este predicado hemos creado la acción `prep-ejercicio`.

Para adaptar la acción `realizar-ejercicio` solo hemos tenido que añadir la precondición `(preparado ?e ?d2)`.

2.3. Extensión 2

En el caso de la segunda extensión los cambios en el dominio han sido mínimos. Solamente hemos tenido que añadir el predicado `(predecesor ?x - ejercicio ?y - ejercicio)` y la precondición `(predecesor ?prev ?e)` a la acción `realizar-ejercicio`.

2.4. Extensión 3

Para esta extensión hemos decidido utilizar una función llamada `(ejercicios-dia ?d - dia)` que contabiliza los ejercicios realizados en el día `?d`. De esta forma solo necesitamos añadir a la acción `realizar-ejercicio` la precondición `(<(ejercicios-dia ?d2) 6)` para controlar que no hayan más de 6 de ejercicios en un mismo día, y el efecto `(increase (ejercicios-dia ?d2) 1)` para ir actualizando el contador.

Para obtener una mayor eficiencia también hemos añadido la precondición `(<(ejercicios-dia ?d2) 6)` a la acción `prep-ejercicio`.

2.5. Extensión 4

Para esta extensión usamos una función `(minutos-ej ?ej - ejercicio)` para guardar la información de el tiempo empleado por cada ejercicio. También añadimos una función `(minutos-ej ?ej - ejercicio)` para contabilizar los minutos de ejercicios realizados para cada día.

De este modo podemos añadir una precondición en la acción de añadir ejercicio garantizando que el tiempo de cada día se mantenga siempre por debajo del limite de 90 minutos.

3. Modelado de los problemas a resolver

3.1. Nivel básico

Para el nivel básico, al igual que para todas las extensiones, los objetos consisten en una lista con todos los ejercicios (`e{i}`), un ejercicio extra llamado `dummy`, una lista de los dias (`d0...d15`) y un listado con todos los niveles del 0 al 10 (`n0...n10`).

Para el estado inicial primero de todo ordenamos los 15 dias utilizando el predicado `(prev ?x - dia ?y - dia)`. Los niveles los ordenamos también. Pero a diferencia de los dias, solo añadimos como sucesor de un nivel el mismo nivel o el inmediatamente superior. (por ejemplo: `(next-nivel n2 n2) (next-nivel n2 n3)`).

Para cada día, le asignamos el ejercicio `dummy` como el ultimo que se ha hecho. Esto sirve para permitir que se cumplan las precondiciones cuando aun no hay ningún ejercicio asignado a un dia. Usando el predicado: `(last ?dia dummy)` para todos los valores de `?dia`.

Consecuentemente tenemos que declarar como hecho el ejercicio `dummy` para cada uno de los dias. Usando el predicado `(hecho dummy ?dia)` para todos los valores de `dia`.

Seguidamente añadimos la relación entre todos los ejercicios preparadores y el respectivo ejercicio que preparan con el predicado `(preparador x y)`. Para los ejercicios que no tienen preparador se les asigna el ejercicio `dummy` como preparador.

Para cada ejercicio que hace el usuario declaramos con el predicado realiza que lo ha realizado en el día `d0`. También declaramos con el predicado `reached` el nivel inicial como

nivel al que se ha llegado en ese ejercicio. Para los ejercicios que no se han realizado se marcan también como realizados en el día 0 pero con nivel 0 y también se les asigna `reached` de 0.

Utilizando el predicado `objetivo` se declaran los objetivos del usuario para cada ejercicio.

En el estado final buscamos que todos objetivos se hayan alcanzado de la siguiente forma:

```
(forall (?ej - ejercicio ?n - nivel)
    (imply (objetivo ?ej ?n) (reached ?ej ?n))
)
```

Para el nivel básico y para todas las extensiones hemos escrito un código en Python que genera el modelo del problema un modo más simple.

3.2. Extensión 1

En esta extensión ya no se declara `dummy` como preparador de los ejercicios que no tienen preparador.

3.3. Extensión 2

En esta extensión se añaden los predecesores. Para ello se usa `(predecesor x y)` para todos los ejercicios con predecesor. Para los ejercicios sin predecesor se declaran como predecesores todos los ejercicios (un ejercicio sin predecesor puede estar precedido por cualquier otro ejercicio).

3.4. Extensión 3

Para la tercera extensión tenemos que inicializar el número de ejercicios de cada día a 0. Esto lo hacemos con `(= (ejercicios-dia d{i}) 0)`.

3.5. Extensión 4

En esta última extensión en vez de inicializar el numero de ejercicios queremos inicializar el número de minutos por día. Esto lo hacemos con `(= (minutos-dia d{i}) 0)`

Por otro lado también tenemos que asignarle a cada ejercicio la cantidad total de minutos que requiere. Para ello utilizamos `(= (minutos-ej e{i}) MINS)`.

4. Desarrollo de modelos

Para el desarrollo de los distintos problemas hemos optado por un diseño incremental basado en prototipos siguiendo el guión del enunciado de la práctica. Empezamos por el nivel básico y se fue ampliando poco a poco, teniendo en cuenta cada una de las extensiones.

En cada prototipo hemos ido añadiendo nuevos elementos del problema hasta llegar a la última extensión.

Las extensiones 3 y 4 se han construido de forma muy similar por lo que se ha podido generar una en base a la otra.

5. Problemas de prueba

Todos los juegos de prueba utilizados se generaron con un script de Python para simplificar el proceso. Por otra parte, como FF no muestra el modelo final, comprobar si es correcto es una tarea muy complicada. Para facilitar esta tarea también escribimos un parse en c++ que generaba una salida más entendible. Para usar el programa tenemos un script run.sh que tiene como primer parámetro la versión (basico, ext1, ext2 ...) y ejecuta el generador correspondiente, el ff y finalmente el parser para visualizar la salida.

5.1. Nivel básico

Hicimos 3 problemas de prueba, el primero sin ningún preparador para comprobar que funcionaba a nivel mas básico. El segundo con algún preparador para comprobar el funcionamiento correcto de los preparadores y el tercero con preparadores de preparadores, para comprobar que funciona en el caso que se forme una cadena de preparadores.

5.1.1. problema de prueba 1

```
;; numero de ejercicios: ;; 7
;; prep: ;;  {'1': ['3'], '3': ['4'], '7': ['5']}
;; haciendo: ;;  {'1': '3', '3': '4', '5': '5', '7': '6'}
;; objetivos: ;;  {'1': '10', '3': '9', '5': '7', '7': '10'}
;; ... (orden de dias, niveles, dummies, etc...)

;; preparadores
(preparador e3 e1)
(preparador e4 e3)
(preparador e5 e7)

(preparador dummy e2) (preparador dummy e4)
(preparador dummy e5) (preparador dummy e6)

;; Ejercicios hechos (realizados el dia 0 con sus respectivos niveles)
(realiza e1 n3 d0) (reached e1 n3)
(realiza e3 n4 d0) (reached e3 n4)
(realiza e5 n5 d0) (reached e5 n5)
(realiza e7 n6 d0) (reached e7 n6)

;; Ejercicios NO hechos (asignamos nivel 0)
(realiza e2 n0 d0) (reached e2 n0)
(realiza e4 n0 d0) (reached e4 n0)
(realiza e6 n0 d0) (reached e6 n0)

;; objetivos
(objetivo e1 n10) (objetivo e3 n9)
(objetivo e5 n7) (objetivo e7 n10)
```

5.1.2. problema de prueba 2

```
;; numero de ejercicios: ;; 9
;; prep: ;;  {'1': ['3'], '3': ['9'], '7': ['5'], '8': ['4']}
;; haciendo: ;;  {'1': '3', '3': '4', '5': '4', '7': '5', '8': '1', '9': '9'}
;; objetivos: ;;  {'1': '10', '3': '9', '5': '7', '7': '10', '8': '3'}
;; ...
;; preparadores
(preparador e3 e1) (preparador e9 e3)
(preparador e5 e7) (preparador e4 e8)

(preparador dummy e2) (preparador dummy e4)
(preparador dummy e5) (preparador dummy e6)
(preparador dummy e9)

;; Ejercicios hechos
(realiza e1 n3 d0) (reached e1 n3)
(realiza e3 n4 d0) (reached e3 n4)
(realiza e5 n4 d0) (reached e5 n4)
(realiza e7 n5 d0) (reached e7 n5)
(realiza e8 n1 d0) (reached e8 n1)
(realiza e9 n9 d0) (reached e9 n9)

;; Ejercicios NO hechos
(realiza e2 n0 d0) (reached e2 n0)
(realiza e4 n0 d0) (reached e4 n0)
(realiza e6 n0 d0) (reached e6 n0)

;; objetivos
(objetivo e1 n10) (objetivo e3 n9)
(objetivo e5 n7) (objetivo e7 n10)
(objetivo e8 n3)
```

5.2. Extensión 1

Se usaron los 3 problemas del nivel básico añadiendo preparadores adicionales para tener ejercicios con mas de un preparador.

5.3. Extensión 2

Se genero un problema en el que solo había predecesores (sin ningún preparador), otro en el que había tanto preparadores como predecesores y otro en el que había también una cadena de predecesores.

5.4. Extensión 3

En el nivel 3 se partió de la los problemas de nivel 2 y se generaron 3 problemas para 3 escenarios distintos: sin predecesores ni preparadores; sin predecesores y con preparadores

y predecesores. Usamos los mismos juegos de prueba que en la extensión anterior pero añadiendo los predicados de predecesores correspondientes.

5.5. Extensión 4

Se usaron los mismos problemas que en la extensión 3 añadiendo la información de tiempo de cada ejercicio.

5.6. Resultados

dominio	problema	tiempo total
basico	algun_preparador	0.42
basico	preparador_de_preparador	0.47
basico	sin_preparadores	0.34
ext1	algun_preparador	0.80
ext1	preparador_de_preparador	5.61
ext2	predecesor_de_predecesor	10.33
ext2	solo_predecesores	0.35
ext2	un_poco_de_todo	1.06
ext3	sin_nada	1.44
ext3	sin_predecesores	5.00
ext3	un_poco_de_todo	4.56
ext4	sin_nada	1.43
ext4	sin_predecesores	4.66
ext4	un_poco_de_todo	2.45

Como podemos observar en la tabla, la mayoría de dominios y problemas tardan relativamente poco en resolverse. Cabe destacar que la extensión 2 (ext2) tarda demasiado tiempo en resolver el problema predecesor_de_predecesor y con entradas aun más complicadas se queda colgado. Obviamente las siguientes extensiones también tardarán demasiado si encuentran predecesores de predecesores en el problema.

Por otro lado podemos ver que la extensión 3 es en general más lenta que la extensión 4, esto probablemente se deba a que los tiempos asignados a cada ejercicio en la extensión 4 sean tan bajos que no supongan una restricción.