UNIVERSITAT POLITÉCNICA DE CATALUNYA

INTELIGENCIA ARTIFICIAL

BACHELOR DEGREE IN COMPUTER SCIENCE

PROYECTO DE PLANIFICACIÓN

Authors:
Daniel Moreno
Fabio De Angelis
Solange Palomino

Professor:
Javier Vazquez Salceda

Q2 Curso 2019-2020

Índice

Introducción

En esta práctica se presenta un problema que será resuelto usando un sistema de planificación automática.

Para la realización de la práctica hemos seguido las siguientes fases:

Identificación del problema	Determinar si el problema es adecuado para desarrollarlo a través de la planificación automática. También consiste en establecer los objetivos del sistema y los resultados esperados.
Formailzación	En esta fase deberemos explicar a detalle la determinación del dominio y el problema
Implementación	En esta fase deberemos construir el sistema de forma incremental, además de informar los desarrollos realizados en cada etapa.
Pruebas	En esta fase debemos escoger casos representativos y resolverlos mediante el sistema. Tendremos una prueba para cada una de las extensiones del problema.

Identificación del problema

Descripción del problema

A modo de extensión del Sistema Basado en Conocimientos desarrollado en la práctica N°2 de la asignatura, se pide desarrollar una herramienta que pueda recomendar un plan de entrenamiento. Este recibirá como input los siguientes elementos:

- La lista de ejercicios que el usuario ha realizado con anterioridad
- Los niveles logrados por cada ejercicio realizado
- La lista de ejercicios que quiere realizar
- Los niveles objetivo a los que el usuario quiere llegar (del 1 al 10) por cada ejercicio
- Los ejercicios que se pueden realizar
- La relación de dependencia de los ejercicios

Una vez que se cuenta con esos datos, el sistema debe ser capaz de organizar los ejercicios mínimos a realizar para que el usuario esté apto de realizar los ejercicios (y en el nivel que desee) a los 15 días de entrenamiento.

Para esto es muy importante tener en cuenta ciertas reglas.

Relación entre ejercicios

Ejercicios precursores	Este tipo de ejercicio tiene como restricción el hecho de que debe realizarse en el mismo día e inmediatamente antes del ejercicio del que es precursor.
Ejercicios preparadores	Este tipo de ejercicio tiene como restricción el hecho de que debe realizarse en el mismo día y antes del ejercicio del que es preparador. Pero pueden haber varios ejercicios en el medio. Sólo es primordial que se haya realizado.
Regla N+1	Esta regla se debe cumplir para todos los ejercicios dentro del sistema y requiere que se puede realizar un ejercicio en el nivel N+1, sólo si ya se ha realizado el mismo ejercicio en el nivel N en algún día previo.

Viabilidad de utilizar la Planificación Automática

Antes de intentar resolver el problema, debemos saber identificar qué tipo de problema es y cómo puede resolverse, teniendo en cuenta cuestiones como: la información del dominio que se tiene, el tamaño, los inputs, cómo es la solución que se ha de ofrecer, entre otros.

Teniendo en cuenta que la planificación automática se centra en la búsqueda de una serie de rutinas a ejecutar para poder obtener un estado solución vamos descartando la búsqueda local. Esto porque definitivamente el dominio de soluciones y de posibles combinaciones se incrementa demasiado como para fiarnos completamente de esta lógica de resolución.

De hecho, se descarta parcialmente ya que debemos tener en cuenta que en el campo del planner fast forward se usa como primera opción la búsqueda local, y en caso no se encuentre directamente se utiliza la planificación en su totalidad.

Entonces. dado que nos encontramos en un ámbito de complejidad amplia, que usa varias herramientas para lograr su objetivo, vemos conveniente la aplicación de la planificación automática.

Objetivos del sistema

Los objetivos que debe cumplir nuestro proyecto de Planificación Automática son los siguientes:

- ➢ Se deben reflejar los ejercicios que mínimamente el usuario debe realizar para poder ser capaz de realizar los ejercicios objetivo en el día 15 de entrenamiento.
- ➢ Se deben reflejar los niveles de los ejercicios a realizar para llegar a los niveles esperados en el día 15 de entrenamiento.
- ➢ Para cada ejercicio el plan indica en qué día se debe realizar.
- ➢ Para todo ejercicio se cumple que sus predecesores deben realizarse inmediatamente antes y en el mismo día.
- ➢ Para todo ejercicio se cumple que sus preparadores deben realizarse antes pero en el mismo día.
- ➢ Para todo ejercicio de nivel N+1 se cumple que el nivel N ya se ha realizado en algún día anterior.

Resultados esperados por el sistema

Se espera que el sistema pueda mostrar una serie de reglas a cumplir para lograr los objetivos del día 15. Estas reglas se mostrarán en el siguiente formato:

Función	Elemento	Objeto 1	Objeto 2
Ejercitar	Correr	Nivel a ejercitar	Dia a ejercitar
Dia_siguiente		Dia actual	Dia a cambiar

Con lo cual se tendrá información que cumpla con los objetivos del sistema definidos en el apartado anterior, que están relacionados a los niveles y los días en los que los ejercicios se realizan.

Formalización

Modelado del dominio

Por el hecho que este proyecto está articulado en varias extensiones que amplían la solución “básica” , y en particular las extensiones 3 y 4 que son mutuamente excluyentes, se han desarrollados tres dominios diferentes. El primero es para la solución básica y las extensiones 1 y 2, el segundo para la extensión 3 y el tercero para la extensión 4.

A continuación, presentaremos los elementos que conforman al dominio de forma detallada para cada una de las extensiones.

Elementos del dominio

De acuerdo a PDDL, el dominio para un sistema de planificación automática se basa en tres aspectos principales: las variables, los predicados y las acciones.

Tal como se ha explicado en teoría se implementa la sección de la definición de variables y predicados básicamente para reducir la cantidad de errores y para incrementar la eficiencia del programa.

Variables

El dominio del planificador lo hemos definido por siete variables organizadas en 3 capas de jerarquía, tal como se muestra en la siguiente imagen:

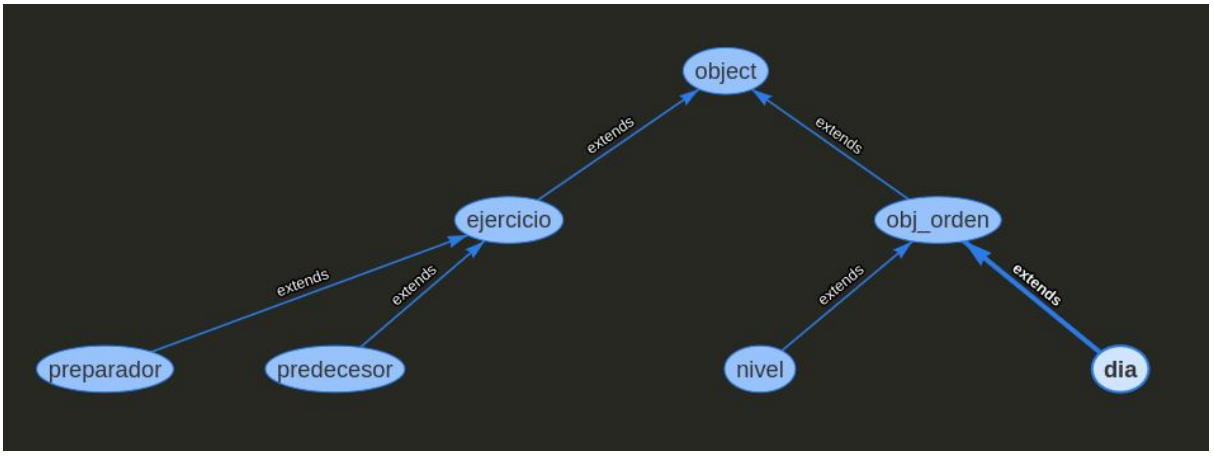

Imagen 1.- Grafo de variables de tipo

En el primer nivel está la variable objeto que engloba todo los datos que serán usados como inputs en el sistema. En el segundo nivel tenemos a la variable ejercicio y el obj_orden que básicamente definimos para poder reutilizar un solo predicado para todas sus variables herederas.

En el tercer nivel tenemos la clasificación de los ejercicios en dos tipos: preparadores y precursores. Esto se da básicamente para poder reconocer las relaciones entre los diversos ejercicios que serán declarados en el problema.

Por otro lado, tenemos al tipo obj_orden que se divide en los sub-tipos dia y nivel. Estos son básicamente los sustantivos que guardan algún tipo de relación de orden como “antes de” para determinar, por ejemplo, que un nivel es previo a otro, por lo que, según el problema, se debe superar antes de proseguir con el siguiente.

Predicados

Los predicados que hemos utilizado para la representación del dominio dentro del problema son los siguientes:

es_preparador	`?x - ejercicio` `?y - ejercicio`	Este predicado requiere de dos parámetros y se utiliza para poder definir la relación entre un ejercicio de tipo preparador y cualquier otro ejercicio.

es_predecesor	`?x - ejercicio` `?y - ejercicio`	Este predicado requiere de dos parámetros y se utiliza para poder definir la relación entre un ejercicio de tipo precursor y cualquier otro ejercicio.
se_ejercita	`?x - ejercicio`	Este predicado requiere de un parámetro y se utiliza para poder definir el hecho de que un ejercicio ya está en proceso de ser realizado. Es decir, que no es nuevo para el usuario.
se_ejercita_en	`?x - ejercicio` `?d - dia`	Este predicado requiere de dos parámetros y se utiliza para poder definir la relación entre un ejercicio cualquiera y el día en el que se debe realizar.
se_ejercita_con	`?x - ejercicio` `?n - nivel`	Este predicado requiere de dos parámetros y se utiliza para poder definir la relación entre un ejercicio cualquiera y el nivel en el que este debe realizarse.
anterior	`?a1 - obj_orden` `?a2 - obj_orden`	Este predicado requiere de dos parámetros y se utiliza para poder definir la relación de orden entre dos obj_orden. En otras palabras, se utiliza para determinar quién va antes que el otro en los objetos de tipo dia y nivel.
ultimo_ejercicio_dia	`?d - dia` `?x - ejercicio`	Este predicado requiere de dos parámetros y se utiliza para poder definir al último ejercicio que se ha cumplido en el día.
hoy	`?d - dia`	Este predicado sólo tiene un parámetro pero nos ayuda a ir avanzando en la organización de los días y así, en la salida, podamos obtener los ejercicios ordenados.

Acciones

Los predicados que hemos utilizado para poder buscar la solución del problema dentro del dominio son los siguientes:

ejercitar ejercicio	?x - ejercicio ?n - nivel ?d - dia	Esta acción nos permite buscar ejercicios para los que el usuario cumple todos los requisitos, por lo que pueden ser realizados.
dia siguiente	?d1 - dia ?d2 - dia	Esta acción nos permite tener un

Funciones

Esta sección sólo se utiliza en las extensiones 3 y 4 del problema. Sin embargo, son importantes ya que nos ayudan plenamente a la optimización de ciertos valores como la cantidad de ejercicios a realizarse en un día y el tiempo total de entrenamiento diario.

Para el uso de estas funciones es necesario incluir los fluentes como un requerimiento del dominio.

Extensión 3	n_ejer_hoy	Esta función nos permite ir manejando la cantidad de ejercicios que están siendo destinados a un determinado día. Ello para que no se supere la cantidad máxima. En el caso base, se deben realizar máximo 6 ejercicios al día.
Extensión 4	t_dia	Esta función nos permite ir manejando el tiempo total que abarcan los ejercicios destinados a un determinado día. Ello para que no se supere el tiempo máximo de entrenamiento. En este caso se debe cubrir máximo 90 minutos al día.
	t_ejer	Esta función nos permite ingresar y conocer el tiempo que requiere un ejercicio determinado para ser realizado.

Elementos del problema

Objetos

Los objetos del problema pertenecen a dos tipos principales: “obj_orden” y “ejercicio”. El primero agrupa todos los objetos necesarios para establecer el orden temporal (día) y la gerarchia de los niveles de dificultad (nivel). El segundo tipo “ejercicio” sirve para para definir los varios ejercicios y los eventuales roles que tienen (preparador o precursor).

El grafo en la imagen 2 muestra en el detalle los varios tipos y subtipos de objetos, y sus jerarquia.

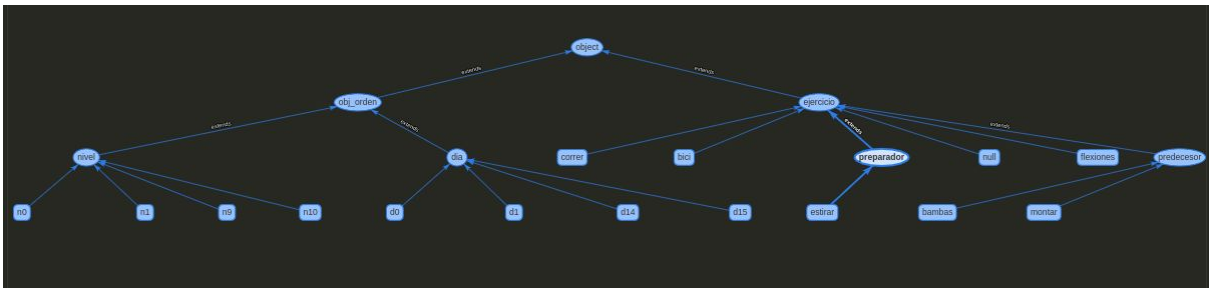

Imagen 2.- Grafo de ejemplo de jerarquía de objetos

Estado inicial

Como el dominio, el estado inicial también requiere de una diferenciación entre las extensiones de la solución. De hecho, para cada uno de los dominios hay un estado inicial diferente.

Los estados iniciales se definen en la sección “:init” de los ficheros de test.

El estado inicial es compuesto de las relaciones entre los ejercicios, es decir, cuáles son los ejercicios preparadores y cuáles son los ejercicios que necesitan un ejercicio precursor.

Además se definen el orden temporal de los días y el orden de dificultad de los niveles, gracias al predicado “anterior”, y los días a los cuales están asignados en principio los ejercicios realizados, gracias al predicados “se_ejercita” y “se_ejercita_en”.

En los siguientes gráficos podemos observar un ejemplo de estado inicial donde se evidencian las relaciones explicadas.

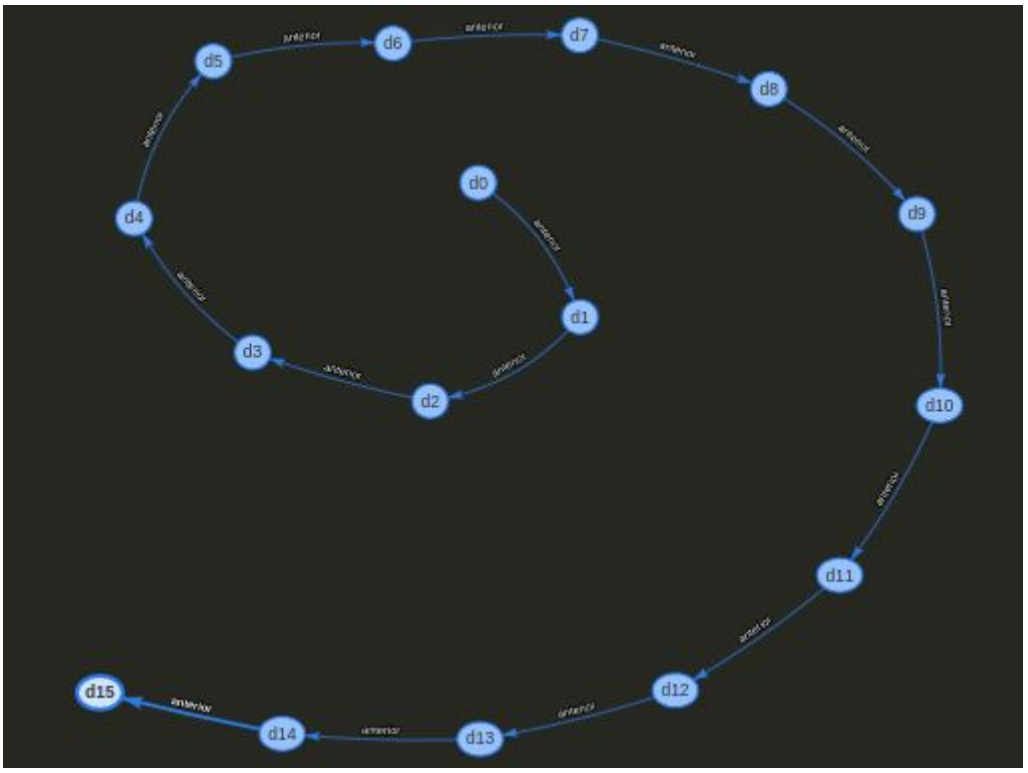

Imagen 3.- Grafo de relación de días

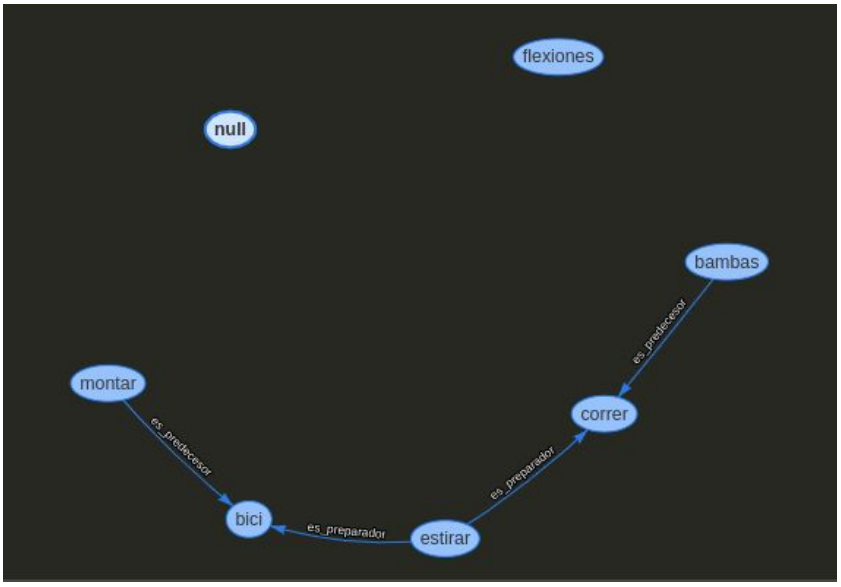

Imagen 4.- Grafo de relación de ejercicios

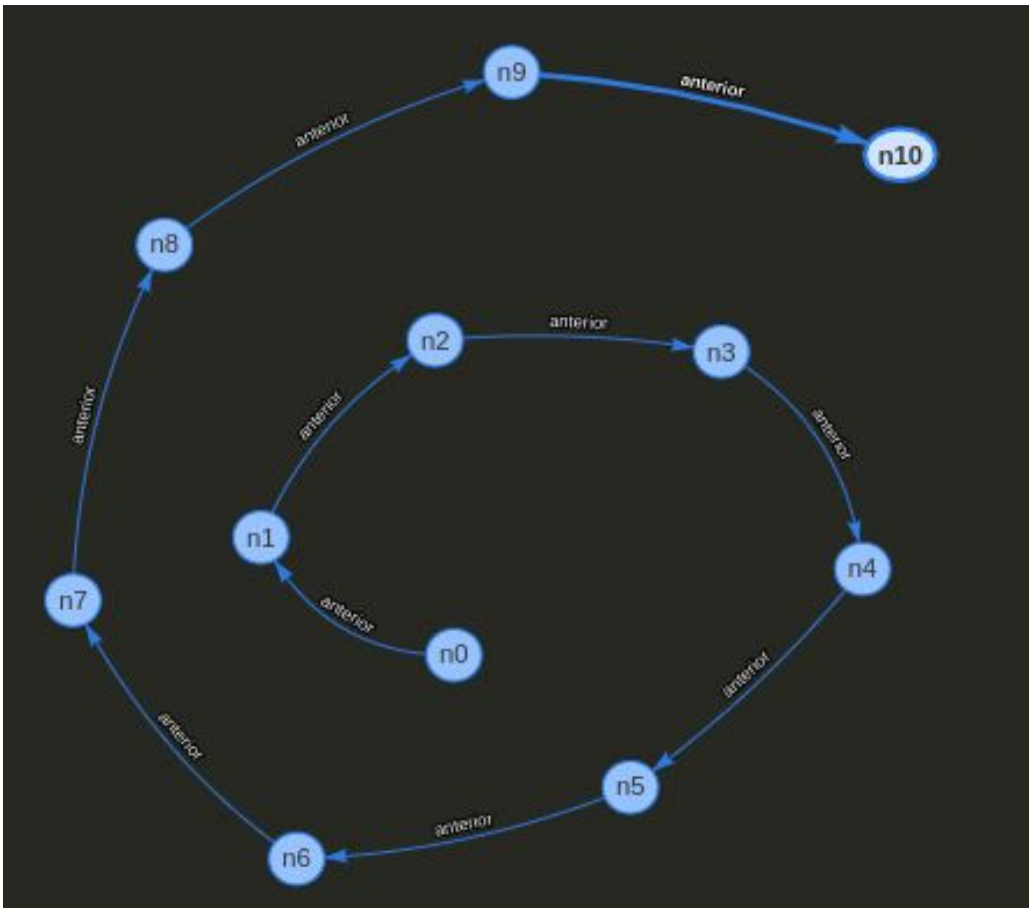

Imagen 5.- Grafo de relación de niveles

Estado final

El estado final describe el objetivo que se quiere lograr, es decir avanzar a los niveles siguientes de los actuales.

Para cada ejercicio se define entonces el nivel objetivo en la sección goal del fichero de test.

```
(:goal (and
            (se_ejercita_con correr n8)
            (se_ejercita_con bici n7)
            (se_ejercita_con flexiones n8)
        )
)
```

Imagen 6.- Representación del objetivo del problema

Implementación

Debido a que el problema mismo estaba planteado de forma escalar, decidimos desarrollar nuestra solución en base a un esquema incremental, en el que íbamos debíamos cumplir una extensión para poder continuar con la otra.

Como primera fase de la implementación del sistema de planificación para el programa de ejercicios, fue importante entender las funcionalidades que se requerían. Es decir, qué información se necesitaba capturar de la solución encontrada. Por lo que consideramos importante plantear una función que nos muestre el día en el que se encuentra el planner, así como los ejercicios que se proponen, además del nivel en el que se debe realizar.

Prototipo 1

En este prototipo se cumplió con el nivel básico, donde básicamente todos los ejercicios tienen entre 0 y 1 ejercicios preparadores. Además se tiene que tener en cuenta que cada ejercicio de nivel N+1 destinado a un día, debe cumplir que se ha realizado el mismo ejercicio con nivel N en algún día anterior.

Prototipo 2

Una vez ya cumpliamos con el nivel básico debíamos probar que este cubra la extensión 1 también, la cual requería que la cantidad de ejercicios preparadores de cada ejercicio podía ser 0, 1 o más de 1. En este caso, nuestro prototipo 1 también funcionaba correctamente. El único cambio que se requería era incrementar la cantidad de preparadores que tenía nuestro problema de prueba.

Prototipo 3

En este prototipo teníamos planeado desarrollar la extensión 2 con la implementación de un objeto “hora_dia” para poder dividir a los días en slots, con los cuales nos ayudaríamos a definir el concepto de “inmediadamente antes” que requerían los ejercicios predecesores. Lo implementamos y funcionaba de forma correcta.

En el mismo sentido, nos proyectamos hacia la extensión 3 y utilizamos al objeto “hora_dia” para poder limitar la cantidad de ejercicios que podían ser asignados a un día. Esta restricción se declararía en el mismo fichero del problema, así sólo se incluirían 6 “slots” para cada día y el planner no podría asignar más ejercicios una vez que todos los slots estaban ocupados.

Asimismo, agregamos los predicados “se_ ejercita_con_h” y “asignado”. El primero era un predicado que requería de tres parámetros: ejercicio, dia, hora_dia; y pretendía almacenar la relación entre un ejercicio y su asignación a una “hora” de un día determinado. En el mismo sentido, el segundo pretendía almacenar el hecho de que un determinado slot ya esté ocupado, por lo que el planner ya no podía usarlo.

No obstante, con este prototipo obteníamos tiempos de ejecución demasiados altos. Además que cuando comenzamos a probar con casos más complejos, nos dimos cuenta de que no se obtenían soluciones. Ante esta situación decidimos regresar al prototipo 2 y plantear una nueva forma de plantear el dominio.

Prototipo 4

En este prototipo quitamos el concepto de “hora_dia” así como los predicados “se_ ejercita_con_h” y “asignado”. Nos propusimos reiniciar la implementación a partir de la extensión 1.

En esta parte del desarrollo incluimos un concepto diferente al anterior denominado “ultimo_ejercicio_día”, el cual nos ayuda a manipular el hecho de que si un ejercicio que queremos ejercitar tiene predecesor y el último ejercicio realizado del dia es su predecesor, entonces se puede asignar el ejercicio a ese día de entrenamiento.

Con ello, luego de hacer muchas pruebas básicas y más complejas, pudimos completar la extensión 2. Asimismo pudimos reducir el tiempo de ejecución de nuestros problemas.

Prototipo 5

Luego de haber desarrollado la extensión 2 en la que básicamente se incluían a los predecesores, debíamos encontrar una forma de restringir el número de ejercicios que se asignan a un día de entrenamiento (ahora sin usar “hora_dia”). Por lo que recurrimos a los fluentes.

De esta manera, declaramos una función denominada “n_ejer_hoy” que incrementamos cada vez que se asignaba un ejercicio a un día. Sin embargo ocurría que debíamos tener en cuenta el cambio de día para que el contador de ejercicios se reinicie.

Así mismo, en los prototipos anteriores, si bien el planner distribuía correctamente los ejercicios en días de entrenamiento, no realizaba la asignación de forma ordenada.

Esto ocasionaba salidas tipo:

```
“ejercitar_ejercicio estirar N2 D2”
“ejercitar_ejercicio correr N2 D2”
“ejercitar_ejercicio bici N2 D2”
“ejercitar_ejercicio estirar N1 D1”
“ejercitar_ejercicio correr N1 D1”
“ejercitar_ejercicio bici N3 D3”
“ejercitar_ejercicio bici N1 D1”
```

Ante ello, recurrimos a incluir el predicado “hoy”, para establecer el día en el que el planner se encuentra; y una acción denominada “dia_siguiente” con el cual nos aseguramos el avance de los días, el orden, y el reinicio del contador de ejercicios asignados. Con lo cual completamos la extensión 3.

Prototipo 6

Teniendo en cuenta que las extensiones 3 y 4 eran mutuamente excluyentes, pero que ya usamos fluentes y al parecer requerían de factores similares. Usamos como base los cambios realizados para la extensión 3 pero adaptados a la extensión 4. Por lo que quitamos la función del contador pero incluimos dos más con el mismo concepto, las cuales denominamos “t_ejer” y “t_dia”.

El primero representa la relación entre un ejercicio y el tiempo que el usuario debe dedicarle en el día. El segundo funciona como un acumulador en el que se almacenan los minutos que se están usando del día que se está analizando. Con este último es que podemos restringir que los minutos diarios destinados al entrenamiento, no superen los 90 minutos, que es el requisito de la extensión 4.

Pruebas

En este apartado mostraremos tres juegos de prueba. Un juego de prueba será para la extensión 2, otro para la extensión 3 y el última para la extensión 4.

Prueba 1

En el primer juego de pruebas utilizaremos como dominio el de la extensión 2. Este dominio nos permite que los ejercicios puedan tener más de un ejercicio preparador y un o ningún ejercicio predecesor.

En este juego de pruebas tenemos tres ejercicios (correr,bici y flexiones), los cuales queremos llegar a nivel 8, 7 y 8, respectivamente. Correr tiene como ejercicio preparador estirar, y como ejercicio predecesor beber. Bici tiene como ejercicio preparador estirar, y como ejercicio predecesor montar_bici .Y finalmente, flexiones tiene como ejercicio preparador estirar. A continuación os mostramos la definición de la instancia de este problema:

```
(define (problem test-01) (:domain dominioEjercicios)
  (:objects   correr bici null flexiones - ejercicio
              d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 d10 - dia
              n0 n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 - nivel
              estirar  - preparador
              beber montar_bici - predecesor )

  (:init
  ;relacion de ejercicios
  (es_preparador estirar bici)
    (es_preparador estirar correr)
 (es_preparador estirar flexiones)

 (es_predecesor beber correr)
    (es_predecesor montar_bici bici)



    ;niveles
    (anterior n0 n1)
```

```
(anterior n1 n2)
(anterior n2 n3)
(anterior n3 n4)
(anterior n4 n5)
(anterior n5 n6)
(anterior n6 n7)
(anterior n7 n8)
(anterior n8 n9)
(anterior n9 n10)

;dias
(anterior d0 d1)
(anterior d1 d2)
(anterior d2 d3)
(anterior d3 d4)
(anterior d4 d5)
(anterior d5 d6)
(anterior d6 d7)
(anterior d7 d8)
(anterior d8 d9)
(anterior d9 d10)

(ultimo_ejercicio_dia d1 null)
(ultimo_ejercicio_dia d2 null)
(ultimo_ejercicio_dia d3 null)
(ultimo_ejercicio_dia d4 null)
(ultimo_ejercicio_dia d5 null)
(ultimo_ejercicio_dia d6 null)
(ultimo_ejercicio_dia d7 null)
(ultimo_ejercicio_dia d8 null)
(ultimo_ejercicio_dia d9 null)
(ultimo_ejercicio_dia d10 null)

;ejercicios realizados
(se_ejercita correr)
(se_ejercita_en correr d0)
(se_ejercita_con correr n0)

(se_ejercita estirar)
```

```
    (se_ejercita_en estirar d0)
    (se_ejercita_con estirar n0)

    (se_ejercita montar_bici)
    (se_ejercita_en montar_bici d0)
    (se_ejercita_con montar_bici n0)

    (se_ejercita beber)
    (se_ejercita_en beber d0)
    (se_ejercita_con beber n0)

    (se_ejercita bici)
    (se_ejercita_en bici d0)
    (se_ejercita_con bici n0)

    (se_ejercita flexiones)
    (se_ejercita_en flexiones d0)
    (se_ejercita_con flexiones n0)

  )

  (:goal (and (se_ejercita_con correr n8) (se_ejercita_con bici n7) (se_ejercita_con flexiones n8)
))
)
```

El plan que obtenemos es el de continuación:

```
step    0: EJERCITAR_EJERCICIO ESTIRAR N1 D1
        1: EJERCITAR_EJERCICIO BEBER N1 D1
        2: EJERCITAR_EJERCICIO ESTIRAR N2 D2
        3: EJERCITAR_EJERCICIO CORRER N1 D1
        4: EJERCITAR_EJERCICIO MONTAR_BICI N1 D1
        5: EJERCITAR_EJERCICIO BEBER N2 D2
        6: EJERCITAR_EJERCICIO BICI N1 D1
        7: EJERCITAR_EJERCICIO FLEXIONES N1 D1
        8: EJERCITAR_EJERCICIO ESTIRAR N3 D3
        9: EJERCITAR_EJERCICIO CORRER N2 D2
        10: EJERCITAR_EJERCICIO MONTAR_BICI N2 D2
```

```
11: EJERCITAR_EJERCICIO BEBER N3 D3
12: EJERCITAR_EJERCICIO BICI N2 D2
13: EJERCITAR_EJERCICIO FLEXIONES N2 D2
14: EJERCITAR_EJERCICIO ESTIRAR N4 D4
15: EJERCITAR_EJERCICIO CORRER N3 D3
16: EJERCITAR_EJERCICIO MONTAR_BICI N3 D3
17: EJERCITAR_EJERCICIO BEBER N4 D4
18: EJERCITAR_EJERCICIO BICI N3 D3
19: EJERCITAR_EJERCICIO FLEXIONES N3 D3
20: EJERCITAR_EJERCICIO ESTIRAR N5 D5
21: EJERCITAR_EJERCICIO CORRER N4 D4
22: EJERCITAR_EJERCICIO MONTAR_BICI N4 D4
23: EJERCITAR_EJERCICIO BEBER N5 D5
24: EJERCITAR_EJERCICIO BICI N4 D4
25: EJERCITAR_EJERCICIO FLEXIONES N4 D4
26: EJERCITAR_EJERCICIO ESTIRAR N6 D6
27: EJERCITAR_EJERCICIO CORRER N5 D5
28: EJERCITAR_EJERCICIO MONTAR_BICI N5 D5
29: EJERCITAR_EJERCICIO BEBER N6 D6
30: EJERCITAR_EJERCICIO BICI N5 D5
31: EJERCITAR_EJERCICIO FLEXIONES N5 D5
32: EJERCITAR_EJERCICIO ESTIRAR N7 D7
33: EJERCITAR_EJERCICIO CORRER N6 D6
34: EJERCITAR_EJERCICIO MONTAR_BICI N6 D6
35: EJERCITAR_EJERCICIO BEBER N7 D7
36: EJERCITAR_EJERCICIO BICI N6 D6
37: EJERCITAR_EJERCICIO FLEXIONES N6 D6
38: EJERCITAR_EJERCICIO ESTIRAR N8 D8
39: EJERCITAR_EJERCICIO CORRER N7 D7
40: EJERCITAR_EJERCICIO MONTAR_BICI N7 D7
41: EJERCITAR_EJERCICIO BEBER N8 D8
42: EJERCITAR_EJERCICIO BICI N7 D7
43: EJERCITAR_EJERCICIO FLEXIONES N7 D7
44: EJERCITAR_EJERCICIO CORRER N8 D8
45: EJERCITAR_EJERCICIO FLEXIONES N8 D8
```

El qual podemos ver que ejercicio se hace cada día y con qué nivel se ejercita. El orden de realización de los ejercicios es el que se muestra por cada día. Podemos ver que se cumplen todas las restricciones y podemos llegar al nivel que queríamos.

Prueba 2

En el segundo juego de pruebas utilizaremos como dominio el de la extensión 3. Este dominio nos permite que los ejercicios puedan tener más de un ejercicio preparador y un o ningún ejercicio predecesor, y también que no haya más de 6 ejercicios cada día.

En este juego de pruebas tenemos cuatro ejercicios (correr,bici, flexiones y abdominales), los cuales queremos llegar a nivel 8, 7, 8 y 5, respectivamente. Correr tiene como ejercicio preparador estirar, y como ejercicio predecesor beber. Bici tiene como ejercicio preparador estirar, y como ejercicio predecesor montar_bici .Y finalmente, flexiones tiene como ejercicio preparador estirar. El ejercicio abdominales también tiene estirar como ejercicio preparador .

A continuación os mostramos la definición de la instancia de este problema:

```
(define (problem test-01) (:domain dominioEjercicios)
  (:objects   correr bici null flexiones - ejercicio
              d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 d10 d11 d12 d13 d14 d15 - dia
              n0 n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 - nivel
              estirar  - preparador
              bambas montar - predecesor )

  (:init
     ;relacion de ejercicios
     (es_preparador estirar bici)
     (es_preparador estirar correr)
     (es_predecesor bambas correr)
     (es_predecesor montar bici)


     ;niveles
     (anterior n0 n1)
     (anterior n1 n2)
     (anterior n2 n3)
     (anterior n3 n4)
     (anterior n4 n5)
     (anterior n5 n6)
     (anterior n6 n7)
```

```
(anterior n7 n8)
(anterior n8 n9)
(anterior n9 n10)

;dias
(anterior d0 d1)
(anterior d1 d2)
(anterior d2 d3)
(anterior d3 d4)
(anterior d4 d5)
(anterior d5 d6)
(anterior d6 d7)
(anterior d7 d8)
(anterior d8 d9)
(anterior d9 d10)
(anterior d10 d11)
(anterior d11 d12)
(anterior d12 d13)
(anterior d13 d14)
(anterior d14 d15)

(ultimo_ejercicio_dia d1 null)
(ultimo_ejercicio_dia d2 null)
(ultimo_ejercicio_dia d3 null)
(ultimo_ejercicio_dia d4 null)
(ultimo_ejercicio_dia d5 null)
(ultimo_ejercicio_dia d6 null)
(ultimo_ejercicio_dia d7 null)
(ultimo_ejercicio_dia d8 null)
(ultimo_ejercicio_dia d9 null)
(ultimo_ejercicio_dia d10 null)
(ultimo_ejercicio_dia d11 null)
(ultimo_ejercicio_dia d12 null)
(ultimo_ejercicio_dia d13 null)
(ultimo_ejercicio_dia d14 null)
(ultimo_ejercicio_dia d15 null)

;ejercicios realizados
(se_ejercita correr)
```

```
    (se_ejercita_en correr d0)
    (se_ejercita_con correr n0)

    (se_ejercita estirar)
    (se_ejercita_en estirar d0)
    (se_ejercita_con estirar n0)

    (se_ejercita montar)
    (se_ejercita_en montar d0)
    (se_ejercita_con montar n0)

    (se_ejercita bambas)
    (se_ejercita_en bambas d0)
    (se_ejercita_con bambas n0)

    (se_ejercita bici)
    (se_ejercita_en bici d0)
    (se_ejercita_con bici n0)

    (se_ejercita flexiones)
    (se_ejercita_en flexiones d0)
    (se_ejercita_con flexiones n0)

    (hoy d1)
    (= (n_ejer_hoy) 0)

  )

  (:goal (and
            (se_ejercita_con correr n8)
            (se_ejercita_con bici n7)
            (se_ejercita_con flexiones n8)
          )
  )
)
```

El plan que obtenemos es el de continuación:

```
step    0: EJERCITAR_EJERCICIO ESTIRAR N1 D1
        1: EJERCITAR_EJERCICIO FLEXIONES N1 D1
        2: EJERCITAR_EJERCICIO MONTAR N1 D1
        3: EJERCITAR_EJERCICIO BICI N1 D1
        4: EJERCITAR_EJERCICIO BAMBAS N1 D1
        5: EJERCITAR_EJERCICIO CORRER N1 D1
        6: DIA_SIGUIENTE D1 D2
        7: EJERCITAR_EJERCICIO ESTIRAR N2 D2
        8: EJERCITAR_EJERCICIO FLEXIONES N2 D2
        9: EJERCITAR_EJERCICIO MONTAR N2 D2
       10: EJERCITAR_EJERCICIO BICI N2 D2
       11: EJERCITAR_EJERCICIO BAMBAS N2 D2
       12: EJERCITAR_EJERCICIO CORRER N2 D2
       13: DIA_SIGUIENTE D2 D3
       14: EJERCITAR_EJERCICIO ESTIRAR N3 D3
       15: EJERCITAR_EJERCICIO FLEXIONES N3 D3
       16: EJERCITAR_EJERCICIO MONTAR N3 D3
       17: EJERCITAR_EJERCICIO BICI N3 D3
       18: EJERCITAR_EJERCICIO BAMBAS N3 D3
       19: EJERCITAR_EJERCICIO CORRER N3 D3
       20: DIA_SIGUIENTE D3 D4
       21: EJERCITAR_EJERCICIO ESTIRAR N4 D4
       22: EJERCITAR_EJERCICIO FLEXIONES N4 D4
       23: EJERCITAR_EJERCICIO MONTAR N4 D4
       24: EJERCITAR_EJERCICIO BICI N4 D4
       25: EJERCITAR_EJERCICIO BAMBAS N4 D4
       26: EJERCITAR_EJERCICIO CORRER N4 D4
       27: DIA_SIGUIENTE D4 D5
       28: EJERCITAR_EJERCICIO ESTIRAR N5 D5
       29: EJERCITAR_EJERCICIO FLEXIONES N5 D5
       30: EJERCITAR_EJERCICIO MONTAR N5 D5
       31: EJERCITAR_EJERCICIO BICI N5 D5
       32: EJERCITAR_EJERCICIO BAMBAS N5 D5
       33: EJERCITAR_EJERCICIO CORRER N5 D5
       34: DIA_SIGUIENTE D5 D6
       35: EJERCITAR_EJERCICIO ESTIRAR N6 D6
       36: EJERCITAR_EJERCICIO FLEXIONES N6 D6
       37: EJERCITAR_EJERCICIO MONTAR N6 D6
       38: EJERCITAR_EJERCICIO BICI N6 D6
       39: EJERCITAR_EJERCICIO BAMBAS N6 D6
       40: EJERCITAR_EJERCICIO CORRER N6 D6
       41: DIA_SIGUIENTE D6 D7
       42: EJERCITAR_EJERCICIO ESTIRAR N7 D7
       43: EJERCITAR_EJERCICIO MONTAR N7 D7
       44: EJERCITAR_EJERCICIO BICI N7 D7
```

```
45: EJERCITAR_EJERCICIO FLEXIONES N7 D7
46: EJERCITAR_EJERCICIO BAMBAS N7 D7
47: EJERCITAR_EJERCICIO CORRER N7 D7
48: DIA_SIGUIENTE D7 D8
49: EJERCITAR_EJERCICIO FLEXIONES N8 D8
50: EJERCITAR_EJERCICIO ESTIRAR N8 D8
51: EJERCITAR_EJERCICIO BAMBAS N8 D8
52: EJERCITAR_EJERCICIO CORRER N8 D8
```

En el qual podemos ver que ejercicio se hace cada día y con qué nivel se ejercita. El orden de realización de los ejercicios es el que se muestra por cada día. Podemos ver que se cumplen todas las restricciones y podemos llegar al nivel que queríamos, como en la prueba anterior. Además el número de ejercicios en cada día cumple la restricción de no ser mayor de 6.

Prueba 3

En el tercer juego de pruebas utilizaremos como dominio el de la extensión 4. Este dominio nos permite que los ejercicios puedan tener más de un ejercicio preparador y un o ningún ejercicio predecesor, y también que no haya más de 90 minutos ejercitados cada día.

En este juego de pruebas tenemos cuatro ejercicios (correr,bici, flexiones), los cuales queremos llegar a nivel 8, 7, y 8, respectivamente. Correr tiene como ejercicio preparador estirar, y como ejercicio predecesor beber. Bici tiene como ejercicio preparador estirar, y como ejercicio predecesor montar_bici. Y finalmente, flexiones tiene como ejercicio preparador estirar.

Además asignamos el tiempo necesario para cada ejercicio.

A continuación os mostramos la definición de la instancia de este problema:

```
(define (problem test-01) (:domain dominioEjercicios)
  (:objects   correr bici null flexiones - ejercicio
              d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 d10 d11 d12 d13 d14 d15 - dia
              n0 n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 - nivel
              estirar  - preparador
              bambas montar - predecesor )

  (:init
     ;relacion de ejercicios
     (es_preparador estirar bici)
```

```
(es_preparador estirar correr)
(es_predecesor bambas correr)
(es_predecesor montar bici)

(= (t_ejer correr) 10)
(= (t_ejer bici) 20)
(= (t_ejer flexiones) 10)
(= (t_ejer estirar) 10)
(= (t_ejer bambas) 15)
(= (t_ejer montar) 10)
(= (t_ejer null) 0)

(= (t_dia) 0)

;niveles
(anterior n0 n1)
(anterior n1 n2)
(anterior n2 n3)
(anterior n3 n4)
(anterior n4 n5)
(anterior n5 n6)
(anterior n6 n7)
(anterior n7 n8)
(anterior n8 n9)
(anterior n9 n10)

;dias
(anterior d0 d1)
(anterior d1 d2)
(anterior d2 d3)
(anterior d3 d4)
(anterior d4 d5)
(anterior d5 d6)
(anterior d6 d7)
(anterior d7 d8)
(anterior d8 d9)
(anterior d9 d10)
(anterior d10 d11)
(anterior d11 d12)
```

```
(anterior d12 d13)
(anterior d13 d14)
(anterior d14 d15)

(ultimo_ejercicio_dia d1 null)
(ultimo_ejercicio_dia d2 null)
(ultimo_ejercicio_dia d3 null)
(ultimo_ejercicio_dia d4 null)
(ultimo_ejercicio_dia d5 null)
(ultimo_ejercicio_dia d6 null)
(ultimo_ejercicio_dia d7 null)
(ultimo_ejercicio_dia d8 null)
(ultimo_ejercicio_dia d9 null)
(ultimo_ejercicio_dia d10 null)
(ultimo_ejercicio_dia d11 null)
(ultimo_ejercicio_dia d12 null)
(ultimo_ejercicio_dia d13 null)
(ultimo_ejercicio_dia d14 null)
(ultimo_ejercicio_dia d15 null)

;ejercicios realizados
(se_ejercita correr)
(se_ejercita_en correr d0)
(se_ejercita_con correr n0)

(se_ejercita estirar)
(se_ejercita_en estirar d0)
(se_ejercita_con estirar n0)

(se_ejercita montar)
(se_ejercita_en montar d0)
(se_ejercita_con montar n0)

(se_ejercita bambas)
(se_ejercita_en bambas d0)
(se_ejercita_con bambas n0)

(se_ejercita bici)
(se_ejercita_en bici d0)
```

```
    (se_ejercita_con bici n0)

    (se_ejercita flexiones)
    (se_ejercita_en flexiones d0)
    (se_ejercita_con flexiones n0)

    (hoy d1)

  )

  (:goal (and (se_ejercita_con correr n8) (se_ejercita_con bici n7) (se_ejercita_con flexiones n8)
)))
```

El plan que obtenemos es el de continuación:

```
step    0: EJERCITAR_EJERCICIO ESTIRAR N1 D1
        1: EJERCITAR_EJERCICIO FLEXIONES N1 D1
        2: EJERCITAR_EJERCICIO MONTAR N1 D1
        3: EJERCITAR_EJERCICIO BICI N1 D1
        4: EJERCITAR_EJERCICIO BAMBAS N1 D1
        5: EJERCITAR_EJERCICIO CORRER N1 D1
        6: DIA_SIGUIENTE D1 D2
        7: EJERCITAR_EJERCICIO ESTIRAR N2 D2
        8: EJERCITAR_EJERCICIO FLEXIONES N2 D2
        9: EJERCITAR_EJERCICIO MONTAR N2 D2
       10: EJERCITAR_EJERCICIO BICI N2 D2
       11: EJERCITAR_EJERCICIO BAMBAS N2 D2
       12: EJERCITAR_EJERCICIO CORRER N2 D2
       13: DIA_SIGUIENTE D2 D3
       14: EJERCITAR_EJERCICIO ESTIRAR N3 D3
       15: EJERCITAR_EJERCICIO FLEXIONES N3 D3
       16: EJERCITAR_EJERCICIO MONTAR N3 D3
       17: EJERCITAR_EJERCICIO BICI N3 D3
       18: EJERCITAR_EJERCICIO BAMBAS N3 D3
       19: EJERCITAR_EJERCICIO CORRER N3 D3
       20: DIA_SIGUIENTE D3 D4
       21: EJERCITAR_EJERCICIO ESTIRAR N4 D4
       22: EJERCITAR_EJERCICIO FLEXIONES N4 D4
       23: EJERCITAR_EJERCICIO MONTAR N4 D4
       24: EJERCITAR_EJERCICIO BICI N4 D4
       25: EJERCITAR_EJERCICIO BAMBAS N4 D4
       26: EJERCITAR_EJERCICIO CORRER N4 D4
```

```
27: DIA_SIGUIENTE D4 D5
28: EJERCITAR_EJERCICIO ESTIRAR N5 D5
29: EJERCITAR_EJERCICIO FLEXIONES N5 D5
30: EJERCITAR_EJERCICIO MONTAR N5 D5
31: EJERCITAR_EJERCICIO BICI N5 D5
32: EJERCITAR_EJERCICIO BAMBAS N5 D5
33: EJERCITAR_EJERCICIO CORRER N5 D5
34: DIA_SIGUIENTE D5 D6
35: EJERCITAR_EJERCICIO ESTIRAR N6 D6
36: EJERCITAR_EJERCICIO FLEXIONES N6 D6
37: EJERCITAR_EJERCICIO MONTAR N6 D6
38: EJERCITAR_EJERCICIO BICI N6 D6
39: EJERCITAR_EJERCICIO BAMBAS N6 D6
40: EJERCITAR_EJERCICIO CORRER N6 D6
41: DIA_SIGUIENTE D6 D7
42: EJERCITAR_EJERCICIO ESTIRAR N7 D7
43: EJERCITAR_EJERCICIO MONTAR N7 D7
44: EJERCITAR_EJERCICIO BICI N7 D7
45: EJERCITAR_EJERCICIO FLEXIONES N7 D7
46: EJERCITAR_EJERCICIO BAMBAS N7 D7
47: EJERCITAR_EJERCICIO CORRER N7 D7
48: DIA_SIGUIENTE D7 D8
49: EJERCITAR_EJERCICIO FLEXIONES N8 D8
50: EJERCITAR_EJERCICIO ESTIRAR N8 D8
51: EJERCITAR_EJERCICIO BAMBAS N8 D8
52: EJERCITAR_EJERCICIO CORRER N8 D8
```

En el cual podemos ver que ejercicio se hace cada día y con qué nivel se ejercita. El orden de realización de los ejercicios es el que se muestra por cada día. Podemos ver que se cumplen todas las restricciones de la solución “básica” y de las extensiones 1, 2 y 4, de hecho el tiempo de entrenamiento total no supera los 90 minutos como se requiere, y entonces podemos llegar al nivel que queríamos.

Traza de resolución

A continuación presentaremos la relación de los dominios y los problemas que hemos desarrollado para poder probar cada una de las funcionalidades del sistema de planificación automática de recomendación de ejercicios.

Problema	Dominio	Test
Nivel Básico	dominioEjercicios_ext1.pddl	problemaEjercicios_ext1.pddl
Extension 1	dominioEjercicios_ext1.pddl	problemaEjercicios_ext1.pddl
Extension 2	dominioEjercicios_ext2.pddl	problemaEjercicios_ext2.pddl
Extension 3	dominioEjercicios_ext3.pddl	problemaEjercicios_ext3.pddl
Extension 4	dominioEjercicios_ext4.pddl	problemaEjercicios_ext4.pddl