Carlos Bergillos, Adrià Cabeza, Roger Vilaseca

June 3, 2019

Contents

1 Introducción **4**

2 Descripción del problema **4**

3 Nivel básico **4**
- 3.1 Dominio . . . 5
  - 3.1.1 Variables . . . 5
  - 3.1.2 Funciones . . . 5
  - 3.1.3 Predicados . . . 5
  - 3.1.4 Acciones . . . 5
- 3.2 Problema . . . 6
  - 3.2.1 Objetos . . . 6
  - 3.2.2 Estado inicial . . . 6
  - 3.2.3 Estado final . . . 6
- 3.3 Juegos de prueba . . . 6

4 Extensión 1 **6**
- 4.1 Dominio . . . 6
  - 4.1.1 Funciones . . . 6
  - 4.1.2 Predicados . . . 6
  - 4.1.3 Acciones . . . 7
- 4.2 Problema . . . 8
  - 4.2.1 Estado inicial . . . 8
  - 4.2.2 Estado final . . . 10
- 4.3 Juegos de prueba . . . 10

5 Extensión 2 **10**
- 5.1 Dominio . . . 10
  - 5.1.1 Funciones . . . 10
  - 5.1.2 Predicados . . . 10
  - 5.1.3 Acciones . . . 10
- 5.2 Problema . . . 10
  - 5.2.1 Estado inicial . . . 10
  - 5.2.2 Estado final . . . 10
- 5.3 Juegos de prueba . . . 10

6 Extensión 3 **10**
6.1 Dominio 10
6.1.1 Funciones 10
6.1.2 Predicados 10
6.1.3 Acciones 10
6.2 Problema 10
6.2.1 Estado inicial 10
6.2.2 Estado final 10
6.3 Juegos de prueba 10

7 Extensión 4 **10**
7.1 Dominio 10
7.1.1 Funciones 10
7.1.2 Predicados 10
7.1.3 Acciones 10
7.2 Problema 10
7.2.1 Estado inicial 10
7.2.2 Estado final 10
7.3 Juegos de prueba 10

8 Conclusiones **10**

Appendices **11**

A Generador de juegos de prueba **11**

1 Introducción

2 Descripción del problema

En esta práctica nos encargaremos de planificar un proyecto de programación de gran envergadura. Deberemos repartir un conjunto de tareas a realizar entre los programadores disponibles.

En concreto, disponemos de T tareas de programación, cada una de ellas tiene un grado de dificultad asignado (de 1 a 3), y una tiempo estimado de realización estimado (en horas).

También disponemos de un conjunto P de programadores. Cada uno de ellos tiene asignado un grado de habilidad (de 1 a 3), que nos indica lo mucho o poco que esta capacitado para resolver las tareas.

No queremos asignar a los programadores tareas mucho más dificiles de lo que para ellos estan capacitados. En concreto, a un programador solo le podremos asignar tareas de como mucho una unidad más de dificultad de lo que nos indique su habilidad. En caso de que sea necesario asignar a un programador una tarea más dificil que su capacidad, la duración de la realización de la tarea se verá incrementada en 2 horas.

Además, todas las tareas deberán ser revisadas, para ello, hará falta una nueva tarea adicional. Esta nueva tarea de revisión será de la misma dificultad que la tarea original. Los programadores tienen tambien asociada una calidad (de 1 a 2). Si la tarea original es realizada por un programador de calidad 1, la nueva tarea de revisión durará 1 hora, si en cambio el programador era de calidad 2, la nueva tarea de revisión durará 2 horas. cada programador también tiene asociada una calidad (de 1 a 2) Para evitar una recursividad infinita, las nuevas tareas de revisión no requerirán a su vez de revisión, y su tiempo de realización no se verá penalizado por la habilidad del programador que la realiza.

3 Nivel básico

En esta primera versión....

3.1 Dominio

3.1.1 Variables

Para la correcta resolución de este problema de planificación hemos visto conveniente trabajar con variables con tipo. En concreto, hemos necesitado 2 tipos, los cuáles hemos llamado **programador** y **tarea**.

- **programador:** Se utilizará para las variables que correspondan a cada programador del conjunto P.
- **tarea:** Se utilizará para las variables que correspondan a cada tarea del conjunto T.

Para las próximas extensiones ya no se requieren más cambios en las variables, esta será la configuración definitiva.

3.1.2 Funciones

- **habilidadProgramador:**
- **dificultadTarea:**

3.1.3 Predicados

- **asignacion:**
- **tareaAsignada:**

3.1.4 Acciones

- **asignar:**

3.2 Problema

3.2.1 Objetos

3.2.2 Estado inicial

3.2.3 Estado final

3.3 Juegos de prueba

4 Extensión 1

En esta extensión, vamos a añadir a nuestro programa la revision de tareas. La revisón de tareas consiste en que cada vez que algun programador realice una tarea, esta deberá ser revisada por otro programador. La dificultad de la revision será la misma que de la tarea y por lo tanto utilizaremos el mismo criterio para asignar un programador, para revisar que en la sección 3.

4.1 Dominio

4.1.1 Funciones

Para esta extensión vamos a utilizar las mismas funciones que en el apartado anterior.

4.1.2 Predicados

- **asignacionTarea:** Equivalente al predicado "asignacion" en la seccion 3.1.3.
- **asignacionRevision:** Predicado utilizado para asignar a una tarea un programador para que este la revise.
- **tareaAsignada:** Igual que en la sección 3.1.3.
- **tareaRevisada:** Predicado para conocer si la tarea en questión ha sido revisada por algún programador.

4.1.3 Acciones

- **asignar:** Esta acción és equivalente a la acción "assignar" explicada en la sección 3.1.4. Donde el único cambio és el cambio de nombre del predicado de "asignacion" a "asignacionTarea".

- **revisar:** El objetivo de esta acción es a partir de las tareas ya asignadas, asignar una revisión a todas la tareas con un programador diferente del que ha tenido asignada la tarea, el cual debe tener habilidad suficientemente grande para poder revisarla.

  Parametros

  - ?p - programador
  - ?t - tarea

  Precondición

  Para poder realizar esta acción se debe cumplir:

  - La tarea debe estar asignada.
  - La tarea no debe estar revisada
  - La habilidad del programador debe ser màs grande o igual que la dificultad de la tarea más uno.
  - El programador que revisa la tarea no puede ser el mismo que el que le ha sido asignada.

  (and (tareaAsignada ?t) (not (tareaRevisada ?t)) (¿= (habilidadProgramador ?p) (- (dificultadTarea ?t) 1)) (not (asignacionTarea ?p ?t)))

  Postcondición

  Esta acción probocará:

  - Un programador será el revisor de la tarea.
  - La tarea estará revisada.

  (and (asignacionRevision ?p ?t) (tareaRevisada ?t))

4.2 Problema

4.2.1 Estado inicial

En este caso el estado inicial serà equivalente al de la sección 3.2.2.

4.2.2 Estado final

El estado final remplazaremos la comprovación de de que todas las tareas estén asignadas, por la comprovación de que todas las tareas estén revisadas.

(forall (?t - tarea) (tareaRevisada ?t))

4.3 Juegos de prueba

5 Extensión 2

5.1 Dominio

5.1.1 Funciones

5.1.2 Predicados

5.1.3 Acciones

5.2 Problema

5.2.1 Estado inicial

5.2.2 Estado final

5.3 Juegos de prueba

6 Extensión 3

6.1 Dominio

6.1.1 Funciones

6.1.2 Predicados

6.1.3 Acciones

6.2 Problema

6.2.1 Estado inicial

6.2.2 Estado final

6.3 Juegos de prueba

7 Extensión 4

7.1 Dominio

7.1.1 Funciones

7.1.2 Predicados

7.1.3 Acciones

7.2 Problema

7.2.1 Estado inicial

7.2.2 Estado final

Appendices

A Generador de juegos de prueba