Intel.ligència Artificial

(Laboratorio)

Práctica de Planificación

Documentación

Curso 2020/2021 2Q

- Jesús Benítez Díaz
- Mauro García Lorenzo
- Javier Rivera Hernández

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Índice:

1.Introducción 5

2.Dominio 5
2.1 Estado inicial y final 5
2.2 Variables 6
2.2.1 Plato 6
2.2.2 Dia 6
2.2.3 TipoPlato 6
2.3 Predicados 6
2.4 Funciones 7
2.4.1 precio-Total: 7
2.4.2 precioPlato (?p - plato): 7
2.4.3 caloriasPlato (?p -plato): 7
2.5 Operadores 7
2.5.1 anadirPlatoObligatorioPrimero 7
2.5.2 anadirPlatoObligatorioSegundo 8
2.5.3 crearMenu 8
2.5.4 tipoPlatoNoConsecutivo 8

3 Juegos de Prueba 9
3.1 Prueba 1 9
3.1.1 Introducción 9
3.1.2 Entrada 9
3.1.3 Salida 10
3.1.4 Resultado 11
3.2 Prueba 2 11
3.2.1 Introducción 11
3.2.2 Entrada 11
3.2.3 Salida 12
3.2.4 Resultado 13
3.3 Prueba 3 13
3.3.1 Introducción 13
3.3.2 Entrada 13
3.3.3 Salida 14
3.3.4 Resultado 15
3.4 Prueba 4 15
3.4.1 Introducción 15
3.4.2 Entrada 15
3.4.3 Salida 16
3.4.4 Resultado 16

3.5 Prueba 5 17
3.5.1 Introducción 17
3.5.2 Entrada 17
3.5.3 Salida 18
3.5.4 Resultado 19
3.6 Prueba 6 19
3.6.1 Introducción 19
3.6.2 Entrada 19
3.6.3 Salida 20
3.6.4 Resultado 21
3.7 Prueba 7 22
3.7.1 Introducción 22
3.7.2 Entrada 22
3.7.3 Salida 23
3.7.4 Resultado 24
3.8 Prueba 8 24
3.8.1 Introducción 24
3.8.2 Entrada 24
3.8.3 Salida 26
3.8.4 Resultado 26
3.9 Prueba 9 26
3.9.1 Introducción 26
3.9.2 Entrada 26
3.9.3 Salida 28
3.9.4 Resultado 29
3.10 Prueba 10 29
3.10.1 Introducción 29
3.10.2 Entrada 29
3.10.3 Salida 31
3.10.4 Resultado 31
3.11 Prueba 11 31
3.11.1 Introducción 31
3.11.2 Entrada 31
3.11.3 Salida 33
3.11.4 Resultado 34
3.12 Prueba 12 34
3.12.1 Introducción 34
3.12.2 Entrada 34
3.12.3 Salida 36
3.12.4 Resultado 37
4. Completado de los niveles 37

4.1 Nivel Básico 37
4.2 Extensión 1 38
4.3 Extensión 2 38
4.4 Extensión 3 40
4.4 Extensión 4 41
4.4 Extensión 5 42

5.Conclusión 43

1.Introducción

Tras el éxito de la aplicación que diseñaba menús, la empresa “Rico Rico” nos ha vuelto a pedir ayuda. El objetivo será diseñar una nueva funcionalidad que permitirá programar menús semanales en bares y restaurantes. Esto lo haremos empleando Fast Forward.

En este proyecto, queremos poder describir e implementar el dominio que se plantea conjuntamente con distintos ejemplos de problemas a través de lenguajes de descripción (PDDL). De manera que podamos familiarizarnos con lenguajes de representación de acciones y se puedan aplicar planificadores modernos para conseguir inferir una solución que cumpla los requisitos que se nos piden.

2.Dominio

2.1 Estado inicial y final

Para todos los problemas el concepto de estado final será el mismo, un estado en el que no existan días sin planificar. Que un día esté planificado dependerá de ciertas condiciones según la extensión. El estado cambia según la extensión.

Nivel Básico: En el nivel básico se empieza con un estado inicial en el que solo conocemos los platos, si son primeros/segundos y las incompatibilidades entre primeros y segundo. Para que un día esté planificado, no pueden haber dos platos incompatibles un mismo día.

Extensión 1: En la extensión 1 tendremos la misma información que en el nivel básico. Para que el estado final sea válido, no se puede utilizar un plato más de una vez a la semana.

Extensión 2: En la extensión 2 tendremos además de la información ya conocida, conocimiento sobre los tipos de platos y que días son consecutivos. No obstante, ahora no puede haber dos días consecutivos con el mismo tipo de plato.

Extensión 3: En esta extensión tendremos además de la información ya conocida, conocimiento sobre los platos que tienen que usarse en ciertos días. Ahora el estado final también debe cumplir que los platos obligatorios se sirvan en el día que se indica en la definición del problema.

Extensión 4: En esta extensión tendremos además de la información ya conocida, conocimiento sobre las calorías que aporta cada plato. Ahora el estado final también debe cumplir que los platos que se sirven cada día tengan entre 1000 y 1500 calorías.

Extensión 5: En esta extensión tendremos además de la información ya conocida, conocimiento sobre el precio de cada plato. El estado final debe cumplir lo mismo que en la extensión 4, y esta vez además tratará de minimizar el precio.

2.2 Variables

2.2.1 Plato

Representa un plato. Es el principal objeto de nuestro dominio, pues será el que tiene que ser repartido a lo largo de los días. Este puede estar relacionado con otros platos de manera que sean incompatibles, en este caso no pueden estar en el mismo menú. Los platos podrán ser identificados como primeros o como segundos, y en la mayoría de las extensiones no se podrá utilizar un plato en dos días diferentes. Cada plato tendrá asociado un precio y número de calorías.

2.2.2 Dia

Este objeto representa la medida del tiempo en que se agrupan los platos. Los días tienen una relación de orden, es decir, pueden tener anteriores y posteriores. Cada día debe tener un primer y un segundo plato. En la extensión 2 no podremos repetir dos días consecutivos el mismo tipo de plato, de los cuales hablaremos a continuación.

2.2.3 TipoPlato

Necesitaremos asociar a cada plato un tipo, y asi poder satisfacer las restricciones de la extensión 2. Entre primeros y segundos no se puede repetir el tipo de plato en dos días consecutivos.

2.3 Predicados

Para modelar los problemas, hemos ideado lo siguientes predicados:

- *platoUsado(Plato)*: Indica que un plato ya se ha usado. Nos es útil para saber que platos no podemos reutilizar.
- *primerPlatoDia(Dia, Plato)*: Indica que primer plato tiene ese dia. Nos es útil para saber que días faltan por planificar.
- *segundoPlatoDia(Dia, Plato)*: Indica que segundo plato tiene ese dia. Nos es útil saber que días faltan por planificar.
- *primerPlato(Plato)*: Indica que el plato es un primero.
- *segundoPlato(Plato)*: Indica que el plato es un segundo.
- *tipoDePlato(TipoPlato)*: Indica el tipo de plato. Nos es útil para saber asegurarnos que no repetimos tipo de plato dos días consecutivos

- *incompatibles(Plato,Plato)*: Indica que dos platos son incompatibles, y por lo tanto no podrán estar en el mismo menú. El primer plato siempre es un primero y el segundo plato es siempre un segundo.
- *consecutivos(Dia, Dia)*: : Indica que el primer día es inmediatamente anterior al segundo día. Nos es útil para asegurarnos que tipos de platos podemos poner en dos días consecutivos.
- *platoObligatorio(Palto, Dia)*: Indica que plato y en qué día debemos servir ese plato.
- *diaObligatorio(Dia)*: Indica el dia que es obligatoria poner un plato en concreto.
- *diaPlanificado(Dia)*: Indica si es el día está planificado. Nos es útil para saber cuando hemos acabado, solo en el caso de que los 5 días de la semana estén planificados hemos acabado.

2.4 Funciones

Las funciones de Metric-FF actúan como variables numéricas en PDDL. Estas devolverán el valor que contienen en ese momento, que pueden ser modificadas en los efectos de los operadores a partir del valor que se le asignan inicialmente en la planificación del problema.

Se pueden asignar estas variables a elementos de un tipo definido en el problema i.e. calorías de un plato.

2.4.1 precio-Total:

Función que devuelve el entero que acumula el precio total de todos los menús y que será la función a minimizar.

2.4.2 precioPlato (?p - plato):

Función que devuelve el precio del plato ?p.

2.4.3 caloriasPlato (?p -plato):

Función que devuelve las calorías del plato ?p.

2.5 Operadores

En cada uno de los operadores tendremos que asegurarnos que la precondición se cumple. Dependiendo de la extensión algunos operadores no se aplicarán y también la precondición puede cambiar.

2.5.1 anadirPlatoObligatorioPrimero

Este operador se usa para asegurarnos que los primeros platos que están marcados como obligatorios en un día concreto, se añadan al menú si o si. La idea es que si el día está marcado como obligatorio, añada el plato y luego busque un segundo para complementar el menú, manteniendo siempre las diferentes restricciones.

Si el operador cumple la precondición se marcará como usados los platos, se marcará como que este día tiene un primero y un segundo y sumaremos los precios de los platos al precio total.

2.5.2 anadirPlatoObligatorioSegundo

Este operador tiene la misma función que el descrito en el subapartado anterior, pero con la diferencia de que el plato que añadirá es un segundo, y luego buscará complementar con un primero que cumpla las restricciones.

2.5.3 crearMenu

En este operador buscamos completar y añadir los platos que no están marcados como obligatorios. Buscaremos un primero y un segundo que cumpla las restricciones según la extensión.

2.5.4 tipoPlatoNoConsecutivo

Este operador es el que por así decirlo confirma a partir de la extensión 2, si un día está correctamente planificado o no. La idea básicamente es que cuando aquellos días que tengan asignado un primero y un segundo, se compruebe si cumplen la condición de la extensión 2. El día se dará como planificado si se cumple la precondición de este operador.

3 Juegos de Prueba

3.1 Prueba 1

Este juego de prueba es para el nivel Básico.

3.1.1 Introducción

El generador ha diseñado un juego de prueba que consta de 12 platos. Además ha creado 4 incompatibilidades distintas entre ellos.

3.1.2 Entrada

```
(define (problem problema1)
     (:domain planificacion)
     (:objects P0 P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 - plato
     d0 d1 d2 d3 d4 - dia
     )

     (:init

          (incompatibles P2 P8)
          (incompatibles P2 P11)
          (incompatibles P4 P11)
          (incompatibles P4 P8)

          (primerPlato P0)
          (primerPlato P1)
          (primerPlato P2)
          (primerPlato P3)
          (primerPlato P4)
          (primerPlato P5)
          (primerPlato P6)

          (segundoPlato P7)
          (segundoPlato P8)
          (segundoPlato P9)
          (segundoPlato P10)
          (segundoPlato P11)

     )

(:goal ( forall (?d - dia) (diaPlanificado ?d) ) )
     )
```

3.1.3 Salida

```
ff: parsing domain file
domain 'PLANIFICACION' defined
 ... done.
ff: parsing problem file
problem 'PROBLEMA1' defined
 ... done.


no metric specified. plan length assumed.

checking for cyclic := effects --- OK.

ff: search configuration is  best-first on 1*g(s) + 5*h(s) where
    metric is  plan length

advancing to distance:    5
                          4
                          3
                          2
                          1
                          0

ff: found legal plan as follows

step    0: CREARMENU P6 P11 D4
        1: CREARMENU P6 P11 D3
        2: CREARMENU P6 P11 D2
        3: CREARMENU P6 P11 D1
        4: CREARMENU P6 P11 D0


time spent:    0.00 seconds instantiating 155 easy, 0 hard action
templates
               0.00 seconds reachability analysis, yielding 10
facts and 155 actions
               0.00 seconds creating final representation with 10
relevant facts, 0 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.00 seconds searching, evaluating 16 states, to a
max depth of 0
               0.00 seconds total time
```

3.1.4 Resultado

Como podemos observar, se ha generado un menú para cada día instanciado que respeta las incompatibilidades introducidas. Al no tener que asegurarse de que no se repitan platos entre días, nuestro programa ha asignado los mismos platos para todos los días.

3.2 Prueba 2

Este juego de prueba se ha creado para el nivel básico.

3.2.1 Introducción

La entrada consta de 11 platos donde, además se han generado 2 incompatibilidades entre ellos.

3.2.2 Entrada

```
(define (problem problema1)
     (:domain planificacion)
     (:objects P0 P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 - plato
     d0 d1 d2 d3 d4 - dia
     )

     (:init

          (incompatibles P2 P7)
          (incompatibles P0 P6)

          (primerPlato P0)
          (primerPlato P1)
          (primerPlato P2)
          (primerPlato P3)
          (primerPlato P4)
          (primerPlato P5)

          (segundoPlato P6)
          (segundoPlato P7)
          (segundoPlato P8)
          (segundoPlato P9)
          (segundoPlato P10)

     )

(:goal ( forall (?d - dia) (diaPlanificado ?d) ) )
     )
```

3.2.3 Salida

```
ff: parsing domain file
domain 'PLANIFICACION' defined
 ... done.
ff: parsing problem file
problem 'PROBLEMA1' defined
 ... done.


no metric specified. plan length assumed.

checking for cyclic := effects --- OK.

ff: search configuration is  best-first on 1*g(s) + 5*h(s) where
    metric is  plan length

advancing to distance:    5
                          4
                          3
                          2
                          1
                          0

ff: found legal plan as follows

step    0: CREARMENU P5 P10 D4
        1: CREARMENU P5 P10 D3
        2: CREARMENU P5 P10 D2
        3: CREARMENU P5 P10 D1
        4: CREARMENU P5 P10 D0


time spent:    0.01 seconds instantiating 140 easy, 0 hard action
templates
               0.00 seconds reachability analysis, yielding 10
facts and 140 actions
               0.00 seconds creating final representation with 10
relevant facts, 0 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.02 seconds searching, evaluating 16 states, to a
max depth of 0
               0.03 seconds total time
```

3.2.4 Resultado

El resultado de esta prueba no nos aporta ningún dato diferenciador del punto anterior, a excepción de que el tiempo de ejecución supera ya las centésimas de segundo.

3.3 Prueba 3

Esta prueba es para la primera extensión.

3.3.1 Introducción

El generador ha creado una entrada que consta de 13 platos y una sola incompatibilidad. Esta extensión no añadirá nada al nivel básico, a excepción de un detalle que comentaremos en la salida.

3.3.2 Entrada

```
(define (problem problema1)
     (:domain planificacion)
     (:objects P0 P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 - plato
     d0 d1 d2 d3 d4 - dia
     )

     (:init

          (incompatibles P1 P7)

          (primerPlato P0)
          (primerPlato P1)
          (primerPlato P2)
          (primerPlato P3)
          (primerPlato P4)
          (primerPlato P5)

          (segundoPlato P6)
          (segundoPlato P7)
          (segundoPlato P8)
          (segundoPlato P9)
          (segundoPlato P10)
          (segundoPlato P11)
          (segundoPlato P12)

     )

(:goal ( forall (?d - dia) (diaPlanificado ?d) ) )
     )
```

3.3.3 Salida

```
ff: parsing domain file
domain 'PLANIFICACION' defined
 ... done.
ff: parsing problem file
problem 'PROBLEMA1' defined
 ... done.


no metric specified. plan length assumed.

checking for cyclic := effects --- OK.

ff: search configuration is  best-first on 1*g(s) + 5*h(s) where
    metric is  plan length

advancing to distance:    5
                          4
                          3
                          2
                          1
                          0

ff: found legal plan as follows

step    0: CREARMENU P0 P12 D0
        1: CREARMENU P1 P11 D1
        2: CREARMENU P2 P10 D2
        3: CREARMENU P3 P9 D3
        4: CREARMENU P4 P8 D4


time spent:    0.00 seconds instantiating 205 easy, 0 hard action
templates
               0.00 seconds reachability analysis, yielding 36
facts and 205 actions
               0.00 seconds creating final representation with 36
relevant facts, 0 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.00 seconds searching, evaluating 412 states, to a
max depth of 0
               0.00 seconds total time
```

3.3.4 Resultado

El programa nos ha devuelto 5 menús donde, a diferencia del código para el nivel básico no se repiten platos 2 veces en la semana.

3.4 Prueba 4

Entrada para la primera extensión.

3.4.1 Introducción

El generador ha creado una entrada que consta de 14 platos y 2 incompatibilidades.

3.4.2 Entrada

```
(define (problem problema1)
      (:domain planificacion)
      (:objects P0 P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 -
plato
      d0 d1 d2 d3 d4 - dia
      )

      (:init

            (incompatibles P6 P11)
            (incompatibles P2 P11)

            (primerPlato P0)
            (primerPlato P1)
            (primerPlato P2)
            (primerPlato P3)
            (primerPlato P4)
            (primerPlato P5)
            (primerPlato P6)

            (segundoPlato P7)
            (segundoPlato P8)
            (segundoPlato P9)
            (segundoPlato P10)
            (segundoPlato P11)
            (segundoPlato P12)
            (segundoPlato P13)

      )

(:goal ( forall (?d - dia) (diaPlanificado ?d) ) )
      )
```

3.4.3 Salida

```
ff: parsing domain file
domain 'PLANIFICACION' defined
 ... done.
ff: parsing problem file
problem 'PROBLEMA1' defined
 ... done.


no metric specified. plan length assumed.

checking for cyclic := effects --- OK.

ff: search configuration is  best-first on 1*g(s) + 5*h(s) where
    metric is  plan length

advancing to distance:    5
                          4
                          3
                          2
                          1
                          0

ff: found legal plan as follows

step    0: CREARMENU P0 P13 D0
        1: CREARMENU P1 P12 D1
        2: CREARMENU P3 P11 D2
        3: CREARMENU P2 P10 D3
        4: CREARMENU P4 P9 D4


time spent:    0.00 seconds instantiating 235 easy, 0 hard action
templates
               0.00 seconds reachability analysis, yielding 38
facts and 235 actions
               0.00 seconds creating final representation with 38
relevant facts, 0 relevant fluents
               0.00 seconds computing LNF
               0.00 seconds building connectivity graph
               0.01 seconds searching, evaluating 482 states, to a
max depth of 0
               0.01 seconds total time
```

3.4.4 Resultado

Nos ha devuelto una vez más un programa que respeta las incompatibilidades introducidas y no repite platos en los días de la semana.

3.5 Prueba 5

Entrada para la extensión número 2.

3.5.1 Introducción

El generador nos ha creado una entrada que consta de 15 platos y 10 tipos de plato distintos. A cada plato le ha asignado un tipo de plato aleatorio entre los instanciados.

3.5.2 Entrada

```
(define (problem problema1)
     (:domain planificacion)
     (:objects P0 P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 -
plato
     d0 d1 d2 d3 d4 - dia
     A B C D E F G H I J - tipoPlato)

     (:init

          (incompatibles P5 P9)
          (incompatibles P4 P9)
          (incompatibles P2 P9)
          (incompatibles P5 P11)
          (incompatibles P0 P12)
          (incompatibles P1 P11)

          (primerPlato P0)
          (primerPlato P1)
          (primerPlato P2)
          (primerPlato P3)
          (primerPlato P4)
          (primerPlato P5)
          (primerPlato P6)

          (segundoPlato P7)
          (segundoPlato P8)
          (segundoPlato P9)
          (segundoPlato P10)
          (segundoPlato P11)
          (segundoPlato P12)
          (segundoPlato P13)
          (segundoPlato P14)

          (tipoDePlato P0 D)
          (tipoDePlato P1 E)
          (tipoDePlato P2 D)
          (tipoDePlato P3 J)
          (tipoDePlato P4 J)
```

```
            (tipoDePlato P5 D)
            (tipoDePlato P6 F)
            (tipoDePlato P7 C)
            (tipoDePlato P8 D)
            (tipoDePlato P9 C)
            (tipoDePlato P10 J)
            (tipoDePlato P11 G)
            (tipoDePlato P12 D)
            (tipoDePlato P13 A)
            (tipoDePlato P14 I)

            (consecutivos d0 d1)
            (consecutivos d1 d2)
            (consecutivos d2 d3)
            (consecutivos d3 d4)

     )

(:goal ( forall (?d - dia) (diaPlanificado ?d) ) )
     )
```

3.5.3 Salida

```
ff: parsing domain file
domain 'PLANIFICACION' defined
 ... done.
ff: parsing problem file
problem 'PROBLEMA1' defined
 ... done.


no metric specified. plan length assumed.

checking for cyclic := effects --- OK.

ff: search configuration is  best-first on 1*g(s) + 5*h(s) where
    metric is  plan length

advancing to distance:   13
                          7
                          5
                          4
                          3
                          2
                          1
                          0

ff: found legal plan as follows
```

```
step    0: CREARMENU P3 P10 D3
        1: CREARMENU P2 P12 D1
        2: CREARMENU P0 P14 D4
        3: CREARMENU P1 P13 D0
        4: CREARMENU P6 P11 D2
        5: TIPOPLATONOCONSECUTIVO P6 P11 P3 P10 P0 P14 D2 D3 D4 J
J
        6: TIPOPLATONOCONSECUTIVO P1 P13 P2 P12 P6 P11 D0 D1 D2 D
J


time spent:    1.88 seconds instantiating 22288690 easy, 0 hard
action templates
               0.52 seconds reachability analysis, yielding 115
facts and 177430 actions
               0.06 seconds creating final representation with 115
relevant facts, 0 relevant fluents
               0.58 seconds computing LNF
               0.17 seconds building connectivity graph
              15.61 seconds searching, evaluating 717 states, to a
max depth of 0
              18.82 seconds total time
```

3.5.4 Resultado

El programa ha respetado, además de las restricciones aplicadas para las anteriores extensiones, una nueva, que hará que no se asignen platos de un mismo tipo en días consecutivos. Remarcar que el tiempo de ejecución en este caso es significativamente superior a los anteriores.

3.6 Prueba 6

Prueba para la segunda extensión.

3.6.1 Introducción

El generador ha diseñado una entrada con 11 platos y 4 incompatibilidades. Además ha repartido 7 tipos de plato entre los instanciados.

3.6.2 Entrada

```
(define (problem problema1)
     (:domain planificacion)
     (:objects P0 P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 - plato
     d0 d1 d2 d3 d4 - dia
     A B C D E F G - tipoPlato)
```

```
     (:init

          (incompatibles P3 P7)
          (incompatibles P3 P10)
          (incompatibles P5 P10)
          (incompatibles P2 P9)

          (primerPlato P0)
          (primerPlato P1)
          (primerPlato P2)
          (primerPlato P3)
          (primerPlato P4)

          (segundoPlato P5)
          (segundoPlato P6)
          (segundoPlato P7)
          (segundoPlato P8)
          (segundoPlato P9)
          (segundoPlato P10)

          (tipoDePlato P0 A)
          (tipoDePlato P1 E)
          (tipoDePlato P2 C)
          (tipoDePlato P3 D)
          (tipoDePlato P4 F)
          (tipoDePlato P5 B)
          (tipoDePlato P6 B)
          (tipoDePlato P7 C)
          (tipoDePlato P8 B)
          (tipoDePlato P9 D)
          (tipoDePlato P10 F)

          (consecutivos d0 d1)
          (consecutivos d1 d2)
          (consecutivos d2 d3)
          (consecutivos d3 d4)

     )

(:goal ( forall (?d - dia) (diaPlanificado ?d) ) )
     )
```

3.6.3 Salida

```
ff: parsing domain file
domain 'PLANIFICACION' defined
 ... done.
ff: parsing problem file
```

```
problem 'PROBLEMA1' defined
 ... done.


no metric specified. plan length assumed.

checking for cyclic := effects --- OK.

ff: search configuration is  best-first on 1*g(s) + 5*h(s) where
    metric is  plan length

advancing to distance:   14
                          6
                          5
                          4
                          3
                          2
                          1
                          0

ff: found legal plan as follows

step    0: CREARMENU P4 P10 D3
        1: CREARMENU P0 P9 D0
        2: CREARMENU P3 P8 D2
        3: CREARMENU P2 P7 D1
        4: TIPOPLATONOCONSECUTIVO P0 P9 P2 P7 P3 P8 D0 D1 D2 C G
        5: CREARMENU P1 P6 D4
        6: TIPOPLATONOCONSECUTIVO P3 P8 P4 P10 P1 P6 D2 D3 D4 F G


time spent:    0.23 seconds instantiating 2847651 easy, 0 hard
action templates
               0.08 seconds reachability analysis, yielding 87
facts and 25335 actions
               0.00 seconds creating final representation with 87
relevant facts, 0 relevant fluents
               0.07 seconds computing LNF
               0.04 seconds building connectivity graph
               0.42 seconds searching, evaluating 327 states, to a
max depth of 0
               0.84 seconds total time
```

3.6.4 Resultado

Además de lo mencionado para el anterior juego de prueba cabe destacar que, en este caso, el tiempo de ejecución ha disminuido considerablemente respecto a este.

3.7 Prueba 7

Juego de prueba para la extensión 3.

3.7.1 Introducción

Para esta extensión, el generador ha incluido un día en el que un plato debe de ser obligatorio.

3.7.2 Entrada

```
(define (problem problema1)
     (:domain planificacion)
     (:objects P0 P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 - plato
     d0 d1 d2 d3 d4 - dia
     A B C D E F G H - tipoPlato)

     (:init

          (incompatibles P5 P9)
          (incompatibles P2 P9)
          (incompatibles P6 P11)
          (incompatibles P1 P10)
          (incompatibles P3 P7)

          (primerPlato P0)
          (primerPlato P1)
          (primerPlato P2)
          (primerPlato P3)
          (primerPlato P4)
          (primerPlato P5)

          (segundoPlato P6)
          (segundoPlato P7)
          (segundoPlato P8)
          (segundoPlato P9)
          (segundoPlato P10)
          (segundoPlato P11)

          (tipoDePlato P0 A)
          (tipoDePlato P1 C)
          (tipoDePlato P2 G)
          (tipoDePlato P3 D)
          (tipoDePlato P4 H)
          (tipoDePlato P5 F)
          (tipoDePlato P6 F)
          (tipoDePlato P7 A)
          (tipoDePlato P8 F)
          (tipoDePlato P9 E)
          (tipoDePlato P10 G)
```

```
            (tipoDePlato P11 F)

            (consecutivos d0 d1)
            (consecutivos d1 d2)
            (consecutivos d2 d3)
            (consecutivos d3 d4)

            (platoObligatorio P6 d4)

            (diaObligatorio d4)

      )

(:goal ( forall (?d - dia) (diaPlanificado ?d) ) )
      )
```

3.7.3 Salida

```
ff: parsing domain file
domain 'PLANIFICACION' defined
 ... done.
ff: parsing problem file
problem 'PROBLEMA1' defined
 ... done.


no metric specified. plan length assumed.

checking for cyclic := effects --- OK.

ff: search configuration is  best-first on 1*g(s) + 5*h(s) where
    metric is  plan length

advancing to distance:   13
                          7
                          5
                          4
                          3
                          2
                          1
                          0

ff: found legal plan as follows

step    0: CREARMENU P2 P10 D3
        1: CREARMENU P1 P7 D2
        2: CREARMENU P0 P9 D0
```

```
        3: CREARMENU P5 P11 D1
        4: TIPOPLATONOCONSECUTIVO P0 P9 P5 P11 P1 P7 D0 D1 D2 F H
        5: ANADIRPLATOOBLIGATORIOSEGUNDO P3 P6 D4
        6: TIPOPLATONOCONSECUTIVO P1 P7 P2 P10 P3 P6 D2 D3 D4 G H


time spent:    0.40 seconds instantiating 4898534 easy, 0 hard
action templates
               0.10 seconds reachability analysis, yielding 89
facts and 32934 actions
               0.02 seconds creating final representation with 89
relevant facts, 0 relevant fluents
               0.09 seconds computing LNF
               0.04 seconds building connectivity graph
               0.84 seconds searching, evaluating 362 states, to a
max depth of 0
               1.49 seconds total time
```

3.7.4 Resultado

Cabe destacar que, además de que la solución cumple con las restricciones mencionadas en anteriores puntos, ésta también tiene en cuenta una nueva: el plato 6 se oferta en el día 4. El tiempo de ejecución sigue sin aumentar en exceso.

3.8 Prueba 8

Juego de prueba para la extensión 3.

3.8.1 Introducción

Se ha diseñado una entrada con 15 platos y 14 tipos de plato distintos. Además se incluye la restricción de que el plato 6 se tiene que ofertar el día 1.

3.8.2 Entrada

```
(define (problem problema1)
     (:domain planificacion)
     (:objects P0 P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 -
plato
     d0 d1 d2 d3 d4 - dia
     A B C D E F G H I J K L M N O - tipoPlato)

     (:init

          (incompatibles P6 P9)
          (incompatibles P2 P8)
          (incompatibles P1 P8)
          (incompatibles P3 P9)
```

```
            (incompatibles P7 P8)
            (incompatibles P4 P11)

            (primerPlato P0)
            (primerPlato P1)
            (primerPlato P2)
            (primerPlato P3)
            (primerPlato P4)
            (primerPlato P5)
            (primerPlato P6)

            (segundoPlato P7)
            (segundoPlato P8)
            (segundoPlato P9)
            (segundoPlato P10)
            (segundoPlato P11)
            (segundoPlato P12)
            (segundoPlato P13)
            (segundoPlato P14)

            (tipoDePlato P0 H)
            (tipoDePlato P1 H)
            (tipoDePlato P2 J)
            (tipoDePlato P3 C)
            (tipoDePlato P4 B)
            (tipoDePlato P5 L)
            (tipoDePlato P6 M)
            (tipoDePlato P7 O)
            (tipoDePlato P8 F)
            (tipoDePlato P9 N)
            (tipoDePlato P10 J)
            (tipoDePlato P11 K)
            (tipoDePlato P12 L)
            (tipoDePlato P13 J)
            (tipoDePlato P14 F)

            (consecutivos d0 d1)
            (consecutivos d1 d2)
            (consecutivos d2 d3)
            (consecutivos d3 d4)

            (platoObligatorio P6 d1)

            (diaObligatorio d1)

      )

(:goal ( forall (?d - dia) (diaPlanificado ?d) ) )
```

```
)
```

3.8.3 Salida

```
ff: parsing domain file
domain 'PLANIFICACION' defined
 ... done.
ff: parsing problem file
problem 'PROBLEMA1' defined
 ... done.


ff: goal can be simplified to FALSE. No plan will solve it
```

3.8.4 Resultado

Como podemos observar, el programa no ha encontrado una solución válida para este conjunto de restricciones. Esto, suponemos, se debe al elevado número de incompatibilidades entre platos que se han generado (6).

3.9 Prueba 9

Entrada para la cuarta extensión.

3.9.1 Introducción

El generador ha diseñado una entrada con 15 platos, donde se han generado 6 incompatibilidades. Destacar la novedad para esta extensión, la cual tiene en cuenta las calorías que aporta cada plato.

3.9.2 Entrada

```
(define (problem problema1)
     (:domain planificacion)
     (:objects P0 P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 -
plato
     d0 d1 d2 d3 d4 - dia
     A B C D E F G H I J K - tipoPlato)

     (:init

          (incompatibles P4 P10)
          (incompatibles P1 P12)
          (incompatibles P7 P12)
          (incompatibles P0 P12)
          (incompatibles P2 P10)
          (incompatibles P6 P13)
```

```
(primerPlato P0)
(primerPlato P1)
(primerPlato P2)
(primerPlato P3)
(primerPlato P4)
(primerPlato P5)
(primerPlato P6)

(segundoPlato P7)
(segundoPlato P8)
(segundoPlato P9)
(segundoPlato P10)
(segundoPlato P11)
(segundoPlato P12)
(segundoPlato P13)
(segundoPlato P14)

(tipoDePlato P0 D)
(tipoDePlato P1 J)
(tipoDePlato P2 A)
(tipoDePlato P3 A)
(tipoDePlato P4 F)
(tipoDePlato P5 I)
(tipoDePlato P6 E)
(tipoDePlato P7 A)
(tipoDePlato P8 J)
(tipoDePlato P9 I)
(tipoDePlato P10 A)
(tipoDePlato P11 G)
(tipoDePlato P12 B)
(tipoDePlato P13 H)
(tipoDePlato P14 D)

(consecutivos d0 d1)
(consecutivos d1 d2)
(consecutivos d2 d3)
(consecutivos d3 d4)

(platoObligatorio P5 d0)
(platoObligatorio P10 d4)

(diaObligatorio d0)
(diaObligatorio d4)

(=(caloriasPlato P0 )586)
(=(caloriasPlato P1 )709)
(=(caloriasPlato P2 )468)
(=(caloriasPlato P3 )546)
```

```
            (=(caloriasPlato P4 )471)
            (=(caloriasPlato P5 )637)
            (=(caloriasPlato P6 )714)
            (=(caloriasPlato P7 )489)
            (=(caloriasPlato P8 )792)
            (=(caloriasPlato P9 )641)
            (=(caloriasPlato P10 )565)
            (=(caloriasPlato P11 )833)
            (=(caloriasPlato P12 )809)
            (=(caloriasPlato P13 )509)
            (=(caloriasPlato P14 )748)

      )

(:goal ( forall (?d - dia) (diaPlanificado ?d) ) )
      )
```

3.9.3 Salida

```
ff: parsing domain file
domain 'PLANIFICACION' defined
 ... done.
ff: parsing problem file
problem 'PROBLEMA1' defined
 ... done.


no metric specified. plan length assumed.

checking for cyclic := effects --- OK.

ff: search configuration is  best-first on 1*g(s) + 5*h(s) where
    metric is  plan length

advancing to distance:   14
                          8
                          7
                          6
                          5
                          4
                          3
                          1
                          0

ff: found legal plan as follows

step    0: CREARMENU P0 P14 D2
        1: ANADIRPLATOOBLIGATORIOPRIMERO P5 P13 D0
```

```
        2: ANADIRPLATOOBLIGATORIOSEGUNDO P3 P10 D4
        3: CREARMENU P2 P12 D1
        4: CREARMENU P6 P7 D1
        5: TIPOPLATONOCONSECUTIVO P5 P13 P2 P7 P0 P14 D0 D1 D2 A K
        6: CREARMENU P1 P9 D3
        7: CREARMENU P4 P8 D3
        8: TIPOPLATONOCONSECUTIVO P0 P14 P1 P8 P3 P10 D2 D3 D4 J K


time spent:    2.08 seconds instantiating 25379410 easy, 0 hard
action templates
               0.52 seconds reachability analysis, yielding 100
facts and 117883 actions
               0.05 seconds creating final representation with 100
relevant facts, 0 relevant fluents
               0.37 seconds computing LNF
               0.11 seconds building connectivity graph
               4.78 seconds searching, evaluating 423 states, to a
max depth of 0
               7.91 seconds total time
```

3.9.4 Resultado

El programa nos ha devuelto una serie de menús que cumplen todas las restricciones de las extensiones anteriores, además de la nueva. En este caso, la suma de las calorías de cada menú no pueden superar en ningún caso las 1500 calorías, ni estar por debajo de 1000.

3.10 Prueba 10

Juego de prueba para la cuarta extensión.

3.10.1 Introducción

La entrada es similar a la de la prueba anterior, con la salvedad de que el número de platos instanciados es inferior. Destacar el elevado número de incompatibilidades (5) para los 10 platos generados.

3.10.2 Entrada

```
(define (problem problema1)
     (:domain planificacion)
     (:objects P0 P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 - plato
     d0 d1 d2 d3 d4 - dia
     A B C D E F G H I J - tipoPlato)

     (:init

          (incompatibles P5 P9)
```

```
(incompatibles P2 P10)
(incompatibles P1 P6)
(incompatibles P1 P9)
(incompatibles P1 P10)

(primerPlato P0)
(primerPlato P1)
(primerPlato P2)
(primerPlato P3)
(primerPlato P4)

(segundoPlato P5)
(segundoPlato P6)
(segundoPlato P7)
(segundoPlato P8)
(segundoPlato P9)
(segundoPlato P10)

(tipoDePlato P0 H)
(tipoDePlato P1 E)
(tipoDePlato P2 G)
(tipoDePlato P3 G)
(tipoDePlato P4 B)
(tipoDePlato P5 G)
(tipoDePlato P6 I)
(tipoDePlato P7 F)
(tipoDePlato P8 G)
(tipoDePlato P9 E)
(tipoDePlato P10 H)

(consecutivos d0 d1)
(consecutivos d1 d2)
(consecutivos d2 d3)
(consecutivos d3 d4)

(platoObligatorio P9 d0)
(platoObligatorio P7 d3)

(diaObligatorio d0)
(diaObligatorio d3)

(=(caloriasPlato P0 )830)
(=(caloriasPlato P1 )871)
(=(caloriasPlato P2 )666)
(=(caloriasPlato P3 )718)
(=(caloriasPlato P4 )692)
(=(caloriasPlato P5 )591)
(=(caloriasPlato P6 )704)
```

```
            (=(caloriasPlato P7 )861)
            (=(caloriasPlato P8 )607)
            (=(caloriasPlato P9 )715)
            (=(caloriasPlato P10 )648)

      )

(:goal ( forall (?d - dia) (diaPlanificado ?d) ) )
      )
```

3.10.3 Salida

```
ff: parsing domain file
domain 'PLANIFICACION' defined
 ... done.
ff: parsing problem file
problem 'PROBLEMA1' defined
 ... done.

ff: goal can be simplified to FALSE. No plan will solve it
```

3.10.4 Resultado

Una vez más, el programa no encuentra solución cuando el número de incompatibilidades crece. Esto combinado con que las restricciones cada vez son mayores, hace que la aplicación sufra.

3.11 Prueba 11

Entrada para la quinta extensión.

3.11.1 Introducción

La entrada ofrecida por el generador cuenta con 12 platos. El número de incompatibilidades es pequeño(1). Además incluye la novedad de esta extensión, en la cual nuestro programa deberá minimizar el precio.

3.11.2 Entrada

```
(define (problem problema1)
      (:domain planificacion)
      (:objects P0 P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 - plato
      d0 d1 d2 d3 d4 - dia
      A B C D E F G H - tipoPlato)

      (:init

            (incompatibles P2 P11)
```

```
(primerPlato P0)
(primerPlato P1)
(primerPlato P2)
(primerPlato P3)
(primerPlato P4)
(primerPlato P5)

(segundoPlato P6)
(segundoPlato P7)
(segundoPlato P8)
(segundoPlato P9)
(segundoPlato P10)
(segundoPlato P11)

(tipoDePlato P0 B)
(tipoDePlato P1 C)
(tipoDePlato P2 C)
(tipoDePlato P3 D)
(tipoDePlato P4 E)
(tipoDePlato P5 G)
(tipoDePlato P6 C)
(tipoDePlato P7 C)
(tipoDePlato P8 E)
(tipoDePlato P9 D)
(tipoDePlato P10 E)
(tipoDePlato P11 E)

(consecutivos d0 d1)
(consecutivos d1 d2)
(consecutivos d2 d3)
(consecutivos d3 d4)

(platoObligatorio P2 d3)

(diaObligatorio d3)

(=(caloriasPlato P0 )819)
(=(caloriasPlato P1 )684)
(=(caloriasPlato P2 )796)
(=(caloriasPlato P3 )876)
(=(caloriasPlato P4 )596)
(=(caloriasPlato P5 )731)
(=(caloriasPlato P6 )596)
(=(caloriasPlato P7 )857)
(=(caloriasPlato P8 )486)
(=(caloriasPlato P9 )805)
(=(caloriasPlato P10 )698)
(=(caloriasPlato P11 )611)
```

```
(= (precio-Total) 0)
           (=(precioPlato P0 )23)
           (=(precioPlato P1 )9)
           (=(precioPlato P2 )25)
           (=(precioPlato P3 )13)
           (=(precioPlato P4 )13)
           (=(precioPlato P5 )11)
           (=(precioPlato P6 )25)
           (=(precioPlato P7 )15)
           (=(precioPlato P8 )20)
           (=(precioPlato P9 )12)
           (=(precioPlato P10 )23)
           (=(precioPlato P11 )17)

     )

(:goal ( forall (?d - dia) (diaPlanificado ?d) ) )
(:metric minimize (precio-Total))    )
```

3.11.3 Salida

```
ff: parsing domain file
domain 'PLANIFICACION' defined
 ... done.
ff: parsing problem file
problem 'PROBLEMA1' defined
 ... done.


metric established (normalized to minimize):
((1.00*[RF0](PRECIO-TOTAL)) - () + 0.00)

checking for cyclic := effects --- OK.

ff: search configuration is  best-first on 1*g(s) + 5*h(s) where
    metric is ((1.00*[RF0](PRECIO-TOTAL)) - () + 0.00)

advancing to distance:   13
                          7
                          6
                          5
                          4
                          3
                          2
                          1
                          0
```

```
ff: found legal plan as follows

step    0: CREARMENU P0 P8 D2
        1: ANADIRPLATOOBLIGATORIOPRIMERO P2 P6 D3
        2: CREARMENU P4 P7 D0
        3: CREARMENU P5 P10 D4
        4: CREARMENU P3 P11 D1
        5: CREARMENU P1 P9 D1
        6: TIPOPLATONOCONSECUTIVO P4 P7 P3 P9 P0 P8 D0 D1 D2 D H
        7: TIPOPLATONOCONSECUTIVO P0 P8 P2 P6 P5 P10 D2 D3 D4 C H


time spent:    0.39 seconds instantiating 4808641 easy, 0 hard
action templates
               0.11 seconds reachability analysis, yielding 88
facts and 24667 actions
               0.00 seconds creating final representation with 88
relevant facts, 1 relevant fluents
               0.07 seconds computing LNF
               0.03 seconds building connectivity graph
               4.43 seconds searching, evaluating 8667 states, to
a max depth of 0
               5.03 seconds total time
```

3.11.4 Resultado

El juego de prueba diseñado por el generador resulta poco significativo para comprobar la mencionada nueva condición de la extensión 5. Esto es debido a que el número de platos es pequeño, por lo que no deja margen al programa para que minimice los precios del menú.

3.12 Prueba 12

Prueba para la extensión 5.

3.12.1 Introducción

La entrada es similar a la anterior, añadiendo tanto platos como incompatibilidades.

3.12.2 Entrada

```
(define (problem problema1)
     (:domain planificacion)
     (:objects P0 P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 - plato
     d0 d1 d2 d3 d4 - dia
     A B C D E F G H - tipoPlato)

     (:init

          (incompatibles P3 P8)
```

```
(incompatibles P6 P12)
(incompatibles P4 P12)
(incompatibles P4 P11)

(primerPlato P0)
(primerPlato P1)
(primerPlato P2)
(primerPlato P3)
(primerPlato P4)
(primerPlato P5)

(segundoPlato P6)
(segundoPlato P7)
(segundoPlato P8)
(segundoPlato P9)
(segundoPlato P10)
(segundoPlato P11)
(segundoPlato P12)

(tipoDePlato P0 E)
(tipoDePlato P1 C)
(tipoDePlato P2 G)
(tipoDePlato P3 F)
(tipoDePlato P4 G)
(tipoDePlato P5 G)
(tipoDePlato P6 E)
(tipoDePlato P7 G)
(tipoDePlato P8 D)
(tipoDePlato P9 F)
(tipoDePlato P10 G)
(tipoDePlato P11 G)
(tipoDePlato P12 D)

(consecutivos d0 d1)
(consecutivos d1 d2)
(consecutivos d2 d3)
(consecutivos d3 d4)

(platoObligatorio P6 d3)
(platoObligatorio P0 d0)

(diaObligatorio d3)
(diaObligatorio d0)

(=(caloriasPlato P0 )556)
(=(caloriasPlato P1 )823)
(=(caloriasPlato P2 )876)
(=(caloriasPlato P3 )815)
```

```
            (=(caloriasPlato P4 )830)
            (=(caloriasPlato P5 )629)
            (=(caloriasPlato P6 )656)
            (=(caloriasPlato P7 )812)
            (=(caloriasPlato P8 )663)
            (=(caloriasPlato P9 )760)
            (=(caloriasPlato P10 )534)
            (=(caloriasPlato P11 )473)
            (=(caloriasPlato P12 )477)

(= (precio-Total) 0)
            (=(precioPlato P0 )30)
            (=(precioPlato P1 )7)
            (=(precioPlato P2 )24)
            (=(precioPlato P3 )24)
            (=(precioPlato P4 )16)
            (=(precioPlato P5 )28)
            (=(precioPlato P6 )15)
            (=(precioPlato P7 )18)
            (=(precioPlato P8 )14)
            (=(precioPlato P9 )20)
            (=(precioPlato P10 )12)
            (=(precioPlato P11 )28)
            (=(precioPlato P12 )20)

     )

(:goal ( forall (?d - dia) (diaPlanificado ?d) ) )
(:metric minimize (precio-Total))    )
```

3.12.3 Salida

```
ff: parsing domain file
domain 'PLANIFICACION' defined
 ... done.
ff: parsing problem file
problem 'PROBLEMA1' defined
 ... done.


no metric specified. plan length assumed.

checking for cyclic := effects --- OK.

ff: search configuration is  best-first on 1*g(s) + 5*h(s) where
    metric is  plan length

best first search space empty! problem proven unsolvable.
```

```
time spent:    0.35 seconds instantiating 4248262 easy, 0 hard
action templates
               0.08 seconds reachability analysis, yielding 83
facts and 18700 actions
               0.01 seconds creating final representation with 83
relevant facts, 0 relevant fluents
               0.07 seconds computing LNF
               0.01 seconds building connectivity graph
               0.00 seconds searching, evaluating 1 states, to a
max depth of 0
               0.52 seconds total time
```

3.12.4 Resultado

Como podemos observar, el programa no pudo ofrecer una solución que cumpliera todas las restricciones.

4. Completado de los niveles

4.1 Nivel Básico

En el nivel básico se nos pedía implementar un programa que fuera capaz de generar un menú semanal. Cada día de la semana debemos tener un menú que conste de un primero y un segundo, respetando que sean compatibles. Para ello hemos creado dos variables:

(:types plato dia)

Esta variable nos sirve para representar los menús de cada día. También hemos creado 4 predicados:

```
(:predicates
    (primerPlato ?nombrePlato - plato)
    (segundoPlato ?nombrePlato - plato)

    (incompatibles ?nombrePlato1 - plato ?nombrePlato2 - plato)
    (diaPlanificado ?dia - dia)
  )
```

Estos predicados que hemos explicado en el apartado 2.3, nos servirán para distribuir un primero y un segundo para cada día de la semana, y con el predicado *incompatibles* aseguramos que no sea incompatibles. Por último hemos creado una acción:

```
(:action crearMenu
   :parameters(?nombrePlato1 - plato ?nombrePlato2 - plato ?diaSemana - dia)
   :precondition(and (not(diaPlanificado ?diaSemana)) (not(incompatibles
?nombrePlato1 ?nombrePlato2)) (primerPlato ?nombrePlato1) (segundoPlato
?nombrePlato2))
   :effect (diaPlanificado ?diaSemana)
   )
```

Esta acción añadirá dos platos que cumplan la única restricción que tenemos de momento y marcará el día como planificado.

4.2 Extensión 1

Para esta extensión se nos pedía ampliar el nivel básico para que el programa fuera capaz de no repetir platos durante la semana. Para esta extensión las variables seguirán siendo las mismas. Sin embargo hemo tenido que añadir un predicado más:

(:predicates
(platoUsado ?nombrePlato - plato)
)

Este nuevo predicado como ya hemos explicado, nos servirá para marcar y descartar aquellos platos que hemos usado algún día de la semana. Por último hemos modificado la acción *crearMenu* vista en el nivel básico:

(:action crearMenu
:parameters(?nombrePlato1 - plato ?nombrePlato2 - plato ?diaSemana - dia)
:precondition(and (not(diaPlanificado ?diaSemana)) (not(incompatibles ?nombrePlato1 ?nombrePlato2)) (not(platoUsado ?nombrePlato1)) (not(platoUsado ?nombrePlato2)) (primerPlato ?nombrePlato1) (segundoPlato ?nombrePlato2))
:effect (and (platoUsado ?nombrePlato1) (platoUsado ?nombrePlato2) (diaPlanificado ?diaSemana))
)

Como precondición hemos añadido que el plato no haya sido usado. En el caso de que se cree el menú, el sistema marcará como usados ambos platos.

4.3 Extensión 2

En esta extensión nos tocará ampliar las restricciones de nuestro menú semanal un poco más. Incorporamos ahora la condición de que no puede haber dos días consecutivos con el mismo tipo de plato. Para ello añadimos una nueva variable:

(:types tipoPlato)

Esta variable nos permite introducir diferentes tipos de platos. Para poder asociar cada plato a su estilo y además cumplir las nuevas restricciones, hemos añadido algunos predicados:

(:predicates
(primerPlatoDia ?d - dia ?nombrePlato - plato)
(segundoPlatoDia ?d - dia ?nombrePlato - plato)
(tipoDePlato ?nombrePlato - plato ?tipo - tipoPlato)
(consecutivos ?dia1 - dia ?dia2 - dia)
)

Estos nuevos predicados nos permiten satisfacer las nuevas peticiones. Tenemos el predicado *consecutivos,* con el que vamos a poder comprobar qué días son consecutivos, tenemos *tipoDePlato*, que nos da información sobre qué tipo de plato es y, por último, *primerPlatoDia* y *segundoPlatoDia* los usaremos para saber que dias tienen asignados primeros y qué días tienen segundo.

Para terminar veamos que nuevas acciones hemos implementado para cubrir la extensión 2:

```
(:action crearMenu
   :parameters(?nombrePlato1 - plato ?nombrePlato2 - plato ?diaSemana - dia)
   :precondition(and (not(diaPlanificado ?diaSemana)) (not(incompatibles
?nombrePlato1 ?nombrePlato2)) (not(platoUsado ?nombrePlato1)) (not(platoUsado
?nombrePlato2)) (primerPlato ?nombrePlato1) (segundoPlato ?nombrePlato2))
   :effect (and (platoUsado ?nombrePlato1) (platoUsado ?nombrePlato2)
(primerPlatoDia ?diaSemana ?nombrePlato1) (segundoPlatoDia ?diaSemana
?nombrePlato2))
   )

   (:action tipoPlatoNoConsecutivo
    :parameters(?primerPlato1 - plato ?segundoPlato1 - plato ?primerPlato2 - plato
?segundoPlato2 - plato ?primerPlato3 - plato ?segundoPlato3 - plato ?dia1 - dia ?dia2 -
dia ?dia3 - dia ?tipoPlato - tipoPlato ?tipoPlato2 - tipoPlato)
    :precondition(and (consecutivos ?dia1 ?dia2) (consecutivos ?dia2 ?dia3)
                (primerPlatoDia ?dia1 ?primerPlato1)
                (segundoPlatoDia ?dia1 ?segundoPlato1)
                (primerPlatoDia ?dia2 ?primerPlato2)
                (segundoPlatoDia ?dia2 ?segundoPlato2)
                (primerPlatoDia ?dia3 ?primerPlato3)
                (segundoPlatoDia ?dia3 ?segundoPlato3)
                (tipoDePlato ?primerPlato2 ?tipoPlato)
                (not (tipoDePlato ?primerPlato3 ?tipoPlato))
                (not (tipoDePlato ?primerPlato1 ?tipoPlato))
                (tipoDePlato ?segundoPlato2 ?tipoPlato)
                (not (tipoDePlato ?segundoPlato3 ?tipoPlato))
                (not (tipoDePlato ?segundoPlato1 ?tipoPlato)) )

    :effect (and (diaPlanificado ?dia1) (diaPlanificado ?dia2) (diaPlanificado ?dia3))
   )
```

La acción crearMenú la hemos modificado para que marque como cierto los predicados de *primerPlatoDia* y *segundoPlatoDia* cuando se cree un menú.

Por otro lado, tenemos una acción totalmente nueva, cuyo funcionamiento hemos explicado en el apartado 2.3. Con ella podremos asegurarnos de que no haya dos días consecutivos con el mismo tipo de plato.

4.4 Extensión 3

Para esta extensión se nos pedía que incorporemos una nueva restricción. Ahora tendremos una serie de platos que tenemos que servir en un determinado día. En las variables no hemos tocado nada, no obstante en los predicados nos hemos visto obligados a incorporar dos nuevos:

```
(:predicates
    (platoObligatorio ?nombrePlato - plato ?dia1 - dia)
    (diaObligatorio ?dia - dia)
)
```

El primer predicado nos indica que día y que plato será obligatorio servir. El segundo predicado simplemente nos indica que días serviremos los platos. Para poder modelar que estos platos estarán sí o sí, implementamos estas dos nuevas acciones:

```
(:action anadirPlatoObligatorioPrimero
    :parameters(?nombrePlato - plato ?nombrePlato2 - plato ?dia1 - dia)
    :precondition (and (not(diaPlanificado ?dia1)) (diaObligatorio ?dia1)
(platoObligatorio ?nombrePlato ?dia1) (not(incompatibles ?nombrePlato
?nombrePlato2))
                (not(platoUsado ?nombrePlato)) (primerPlato ?nombrePlato)
               (not(platoUsado ?nombrePlato2)) (segundoPlato ?nombrePlato2))
    :effect (and (platoUsado ?nombrePlato) (platoUsado ?nombrePlato2)
(primerPlatoDia ?dia1 ?nombrePlato) (segundoPlatoDia ?dia1 ?nombrePlato2))
   )

   (:action anadirPlatoObligatorioSegundo
    :parameters(?nombrePlato - plato ?nombrePlato2 - plato ?dia1 - dia)
    :precondition (and (not(diaPlanificado ?dia1)) (diaObligatorio ?dia1)
(platoObligatorio ?nombrePlato2 ?dia1) (not(incompatibles ?nombrePlato
?nombrePlato2))
               (not(platoUsado ?nombrePlato)) (primerPlato ?nombrePlato)
               (not(platoUsado ?nombrePlato2)) (segundoPlato ?nombrePlato2))
    :effect (and (platoUsado ?nombrePlato) (platoUsado ?nombrePlato2)
(primerPlatoDia ?dia1 ?nombrePlato) (segundoPlatoDia ?dia1 ?nombrePlato2))
   )
```

Tal y como comentamos en el apartado 2.4, tenemos una acción para incorporar los platos obligatorios que son un primero, y otra acción que añade los segundos.

```
(:action crearMenu
   :parameters(?nombrePlato1 - plato ?nombrePlato2 - plato ?diaSemana - dia)
```

:precondition(and (not(diaPlanificado ?diaSemana)) (not (diaObligatorio ?diaSemana))
(not(incompatibles ?nombrePlato1 ?nombrePlato2)) (not(platoUsado ?nombrePlato1))
(not(platoUsado ?nombrePlato2)) (primerPlato ?nombrePlato1) (segundoPlato ?nombrePlato2))
:effect (and (platoUsado ?nombrePlato1) (platoUsado ?nombrePlato2) (primerPlatoDia ?diaSemana ?nombrePlato1) (segundoPlatoDia ?diaSemana ?nombrePlato2))
)

En la acción de *crearMenu* hemos añadido en la precondición que no pueda tratar los platos obligatorios y así asegurarnos que esos platos no se programan para otro dia.

4.4 Extensión 4

En esta extensión tendremos que controlar las calorías que aportan los platos en un día. Para ellos incluimos en nuestro dominio una función que controle las calorías del plato:

(:functions
(caloriasPlato ?c - plato)
)

Usaremos esta función para saber cuántas calorías aporta cada plato y controlar antes de añadir un plato si estamos en el intervalo de caloría permitido. Para modelar esto hemos modificado las precondiciones de las acciones: *anadirPlatoObligatorioPrimero*, *anadirPlatoObligatorioSegundo*, *crearMenu.* Hemos añadido la siguiente condición:

(>= (+ (caloriasPlato ?nombrePlato1) (caloriasPlato ?nombrePlato2)) 1000) (<= (+ (caloriasPlato ?nombrePlato1) (caloriasPlato ?nombrePlato2)) 1500))

4.4 Extensión 5

Para la última extensión se nos pide que tratemos de minimizar el precio del menú utilizado utilizando Metric-FF. Para esto hemos añadido dos funciones:

(:functions
(precio-Total)
(precioPlato ?p - plato)
)
Con estas dos funciones buscamos saber el precio de cada plato y controlar el precio total del menú con el fin de minimizarlo.

Respecto a las acciones hemos tenido que cambiar los effects de algunas. En *anadirPlatoObligatorioPrimero*, *anadirPlatoObligatorioSegundo* y *crearMenu,* añadimos lo siguiente:

(increase (precio-Total) (+ (precioPlato ?nombrePlato1) (precioPlato ?nombrePlato2)))

Con esto controlamos en todo momento el precio del menú actual. Por último en la definición debemos añadir "**(:metric minimize (precio-Total))"** y así indicarle al programa que debe minimizar el precio total.

5.Conclusión

Nuestro programa se ha mostrado bastante solvente para el nivel básico y las 4 primeras extensiones, mostrando algunas debilidades cuando la proporción de incompatibilidades por número de platos aumenta.

Mencionar también que el código que hemos implementado muestra debilidad a la hora de enfrentar la quinta extensión. Este no encuentra solución en la mayoría de los casos y, cuando lo hace, el coste de ejecución es grande.
Esto, suponemos, es debido al elevado número de restricciones a las que la aplicación se tiene que enfrentar combinadas con el coste de ejecución provocado por la minimización del precio.

El generador de entradas es bastante versátil, aunque no ofrece toda la diversidad de entradas que deseábamos.