Práctica de planificación

Raluca Vijulie Martí Rosell Víctor Alcázar

5 de junio de 2017

Índice

1. Descripción del dominio

1.1. Tipos

Los tipos con los que contamos en el dominio son:

`dia - object`**:** Tipo dia, sus instancias representan dias de la semana, por ejemplo `lunes`.

`plato - object`**:** Tipo plato, sus instancias representan platos de comida, por ejemplo `paella`.

`primero - plato`**:** Tipo primero, hereda de plato. Sus instancias representan platos que se consideran primeros, por ejemplo `ensalada`.

`segundo - plato`**:** Tipo segundo, hereda de plato. Sus instancias representan platos que se consideran segundos, por ejemplo `entrecot`.

`tipo_plato - object`**:** Tipo tipo de plato, sus instancias representan tipos de los que un plato puede ser, por ejemplo `pescado`.

1.2. Predicados

Los predicados con los que contamos en el dominio son:

`(incompatible ?p - primero ?s - segundo)`: Expresa que el primer plato `p` es incompatible con el segundo `s`. Por ejemplo, `(incompatible paella lubina)` indica que `paella` y `lubina` no pueden ser servidas en el mismo dia `d` como primero y segundo respectivamente.

`(preparado ?p - plato)`: Expresa que el plato `p` ya ha sido preparado en esta semana. Por ejemplo, `(preparado paella)` indica que `paella` ya ha sido preparado esta semana.

`(consecutivo ?d1 - dia ?d2 - dia)`: Expresa que el dia `d1` precede al dia `d2`. Por ejemplo, `(consecutivo lunes martes)` indica que `martes` es consecutivo a `lunes`.

`(ultimo_dia ?d - dia)`: Expresa que el dia `d` es el último dia de la semana. Por ejemplo, `(ultimo_dia viernes)` indica que `viernes` es el último dia de la semana.

`(es_de_tipo ?p - plato ?t - tipo_plato)`: Expresa que el plato `p` es de tipo `t`. Por ejemplo, `(es_de_tipo paella pescado)` indica que el plato `paella` es de tipo `pescado`.

`(tipo_dia_primero ?d - dia ?t - tipo_plato)`: Expresa que en el dia `d` se ha servido un primer plato de tipo `t`. Por ejemplo, `(tipo_dia_primero jueves pescado)` indica que en el dia `jueves` se ha servido un primer plato de tipo `pescado`.

`(tipo_dia_segundo ?d - dia ?t - tipo_plato)`: Expresa que en el dia `d` se ha servido un segundo plato de tipo `t`. Por ejemplo, `(tipo_dia_segundo martes sopa)` indica que en el dia `martes` se ha servido un segundo plato de tipo `sopa`.

`(servido_primero ?d - dia)`: Expresa que se ha servido un primer plato el dia `d`. Por ejemplo, `(servido_primero jueves)` indica que se ha servido un primer plato el dia `jueves`.

`(servido_segundo ?d - dia)`: Expresa que se ha servido un segundo plato el dia `d`. Por ejemplo, `(servido_segundo martes)` indica que se ha servido un segundo plato el dia `martes`.

`(menu_primero ?d - dia ?p - primero)`: Expresa que se ha servido un primer plato `p` el dia `d`. Por ejemplo, `(menu_primero jueves paella)` indica que se ha servido un primer plato `paella` el dia `jueves`.

`(menu_segundo ?d - dia ?s - segundo)`: Expresa que se ha servido un segundo plato `p` el dia `d`. Por ejemplo, `(menu_segundo martes fabada)` indica que se ha servido un segundo plato `fabada` el dia `martes`.

1.3. Funciones

`(calorias ?p - plato)`: Función que guarda el valor calórico del plato `p`.

`(precio_total)`: Función que guarda el precio total del menú semanal.

`(precio_plato ?p - plato)`: Función que guarda el precio de un plato `p`.

1.4. Acciones

`servir_ultimo_dia_primero`: De manera resumida, esta accion sirve el primer plato del último dia de la semana.

Esta accion se puede ejecutar bajo la condicion de que

- El primer plato del último dia de la semana no haya sido servido

provocando que

- El primer plato del último dia de la semana sea servido
- El precio total del menu semanal se vea incrementado por el precio del plato a servir

`servir_ultimo_dia_segundo`: De manera resumida, esta accion sirve el segundo plato del último dia de la semana.

Esta accion se puede ejecutar bajo la condicion de que

- El segundo plato del último dia de la semana no haya sido servido

provocando que

- El segundo plato del último dia de la semana sea servido
- El precio total del menu semanal se vea incrementado por el precio del plato a servir

`servir_primero_solo`: De manera resumida, esta accion sirve el primer plato de un dia de la semana cuando su segundo ya ha sido servido.

Esta accion se puede ejecutar bajo la condicion de que

- Ni el primer ni el segundo plato del dia a servir hayan sido servidos
- El plato a servir no haya sido servido en la semana actual
- El primer plato servido el dia consecutivo al dia a servir sea de un tipo distinto del del plato a servir

provocando que

- El primer plato del dia a servir sea servido
- El precio total del menu semanal se vea incrementado por el precio del plato a servir

`servir_primero`: De manera resumida, esta accion sirve el primer plato de un dia de la semana cuando su segundo aún no ha sido servido.

Esta accion se puede ejecutar bajo las condiciones de que:

- El primer plato del dia a servir no haya sido servido
- El segundo plato del dia a servir haya sido servido
- El plato a servir no haya sido servido en la semana actual
- El primer plato servido el dia consecutivo al dia a servir sea de un tipo distinto del del plato a servir
- El plato a servir y el segundo plato del dia a servir no sean incompatibles
- La suma de las calorias del plato a servir y del segundo plato del dia a servir no debe ser menor a 1000 ni mayor a 1500

provocando que:

- El primer plato del dia a servir sea servido
- El precio total del menu semanal se vea incrementado por el precio del plato a servir

`servir_segundo`: De manera resumida, esta accion sirve el segundo plato de un dia de la semana cuando su primero aún no ha sido servido.

Esta accion se puede ejecutar bajo las condiciones de que:

- El segundo plato del dia a servir no haya sido servido
- El primer plato del dia a servir haya sido servido
- El plato a servir no haya sido servido en la semana actual
- El segundo plato servido el dia consecutivo al dia a servir sea de un tipo distinto del del plato a servir
- El plato a servir y el primer plato del dia a servir no sean incompatibles
- La suma de las calorias del plato a servir y del primer plato del dia a servir no debe ser menor a 1000 ni mayor a 1500

provocando que:

- El segundo plato del dia a servir sea servido
- El precio total del menu semanal se vea incrementado por el precio del plato a servir

2. Descripción del modelado de los problemas

2.1. Objetos

Los objetos que declaramos al inicio de un problema son, en función de su tipo:

Tipo `primero`**:** Todos los objetos que son primer plato como por ejemplo, `paella`.

Tipo `segundo`**:** Todos los objetos que son segundo plato como por ejemplo, `entrecot`.

Tipo `dia`**:** Todos los objetos que son un dia de la semana como por ejemplo, `lunes`.

2.2. Estado inicial

El estado inicial se compone de una serie de inicializaciones de predicados y funciones.

Inicialización de predicados En la inicialización, se inicializa un subconjunto de los predicados del dominio. Enunciados a continuación, para cada uno de ellos se declara lo siguiente:

- Los dias que son consecutivos a otros mediante el predicado `consecutivo`. Todos los dias deben tener un consecutivo excepto el último día.
- Cuál es el ultimo dia de la semana mediante el predicado `ultimo_dia`. Es necesario que como mínimo uno de los dias de la semana sea el último.
- Los platos que son incompatibles entre ellos mediante el predicado `incompatible`. No es necesario inicializar este predicado para que el programa se ejecute correctamente.
- De que tipo es cada plato mediante el predicado `es_de_tipo`. Es necesario que todos los platos tengan un tipo.
- Si se quiere fijar un primer plato a un dia de la semana, es necesario instanciar los siguientes predicados para el dia deseado `dia`, el plato deseado `plato` y el tipo del plato deseado `tipo`.

  - `(servido_primero dia)`
  - `(menu_primero dia plato)`
  - `(tipo_dia_primero dia tipo)`
  - `(preparado plato)`

  Análogamente, para fijar un segundo:

  - `(servido_segundo dia)`
  - `(menu_segundo dia plato)`
  - `(tipo_dia_segundo dia tipo)`
  - `(preparado plato)`

Inicialización de funciones De forma similar, se inicializan algunas funciones del dominio en las cuales se declara lo siguiente:

- El numero de calorias que tiene un plato con la función `calorias`. Es necesario que todos los platos tengan un numero de calorias asociado.
- El precio de un plato con la función `precio_plato`. Es necesario que todos los platos tengan un precio asociado.
- El precio total del menú semanal con la función `precio_total`. Esta funcion se debe inicializar a 0 para evitar un comportamiento inesperado.

2.3. Estado final

El estado final del problema es definido por el goal:

```
(:goal (forall (?d - dia)
               (and (servido_primero ?d)
                    (servido_segundo ?d)
                    )
               )
       )
```

lo que indica que el problema se dará por resuelto una vez todos los dias de la semana tengan servidos tanto su primero como segundo plato.

Adicionalmente, usando `metric`, indicamos que queremos minimizar la suma total de los precios de los menues de la semana:

```
(:metric
  minimize (precio_total)
  )
```

3. Descripción del desarrollo de los modelos

Los modelos han sido desarrollados iterativamente. Para cada extensión del dominio se ha creado un modelo distinto que comprueba el subconjunto de predicados que esa extensión necesitaba. Eso ha ayudado a garantizar que el código fuese correcto tras cada extensión implementada.

Cabe destacar que en el paso de la extensión 1 a la 2 precisamos de un cambio total del dominio, lo que provocó a su vez un cambio grande en el modelo.

Si no tenemos en cuenta ese cambio, podemos concluir que los modelos se han ido desarrollando incrementalmente, donde cada versión supone un superconjunto de la anterior, al igual que el dominio.