Trabajo Práctico 2: Programación Lógica

Paradigmas de Lenguajes de Programación — 1er cuat. 2017

Fecha de entrega: 6 de Junio de 2017.

1. Introducción

Deseamos implementar en Prolog un asistente para un juego en el cual los jugadores deben combinar de manera adecuada las herramientas que poseen en su mochila.
Cada herramienta básica será representada usando constantes, como por ejemplo, encendedor, rayo, volatilizador, etc. Cada herramienta tiene un potencial asociado, que por simplicidad asumiremos será un número entero. Vamos a describir el potencial de las herramientas básicas utilizando el predicado binario herramienta. Por ejemplo,

```
herramienta(rayo, 10).
herramienta(volatilizador, 40).
herramienta(encendedor, 5).
```

Para representar herramientas compuestas utilizaremos símbolos de función. Por ejemplo, usaremos el símbolo de función `binaria` para representar a la composición de dos herramientas básicas. Es decir, el término `binaria(rayo,volatilizador)` representa a la combinación de un `rayo` y un `volatilizador`. Cada composición de herramientas tiene asociados un costo y un potencial, que en general se definen en término de de los costos y potenciales de sus componentes.
Usaremos un predicado ternario `composicion(+Composicion, ?Potencial, ?Costo)` para definir las composiciones válidas, su potencial y su costo. Por ejemplo, para definir la composición binaria que excluye la posibilidad de usar un encendedor como primera componente damos la siguiente regla:

```
composicion(binaria(X,Y),P,5):-
      herramienta(X, PX), herramienta(Y,PY), X\= encendedor, P is 2*PX + PY.
```

Ejercicio 1

Definir un operador binario de composición llamado `jerarquica`, similar a `binaria`, pero que permite realizar combinaciones tanto de herramientas básicas como de herramientas compuestas. Por ejemplo, podríamos definir combinaciones como
`jerarquica(jerarquica(binaria(rayo,volatilizador), rayo), rayo)`.
El costo de una composición jerárquica se calcula como el doble del costo asociado a sus dos componentes y se asume que el costo de una componente básica es 1. El potencial se calcula como el producto de los potenciales de sus dos componentes.
La consulta

```
composicion(jerarquica(jerarquica(binaria(rayo,volatilizador), rayo), rayo),X,Y).
```

Tiene como única solución

```
X = 6000,
Y = 26
```

Ejercicio 2

El contenido de una mochila será representado como una lista de herramientas elementales. Por ejemplo, una mochila con dos rayos y un volatilizador puede ser descripta como la lista `[rayo, volatilizador, rayo]`. Definir un predicado cuaternario
`configuracion(+M, ?Conf, ?P, ?C)`
que es verdadero cuando `Conf` es una composición válida de todos los elementos que contiene la mochila `M`, cuyo potencial es `P` y cuyo costo es `C`.
Por ejemplo, la consulta:
`configuracion([rayo,volatilizador,rayo],Conf,P,C).`
tiene, entre otras, las siguientes soluciones:

```
Conf = jerarquica(volatilizador, binaria(rayo, rayo)),
P = 1200,
C = 12

Conf = jerarquica(volatilizador, jerarquica(rayo, rayo)),
P = 4000,
C = 10
```

Notar que la mochila es un multiconjunto donde no importa el orden, pero sí las repeticiones. Una configuración es válida si cada una de las herramientas básicas aparece en la configuración tantas veces como en la mochila. Mientras la consulta
`configuracion([rayo,rayo], binaria(rayo,rayo), 30, 5).`
tiene éxito, la consulta
`configuracion([rayo], binaria(rayo,rayo), 30, 5).`
no lo tiene.

Ejercicio 3

Una mochila `M1` es más potente que otra `M2` si permite armar una configuración cuyo potencial sea mayor al de todas las configuraciones admitidas por `M2`.
Definir el predicado binario `masPoderosa(+M1,+M2)` que es verdadero cuando `M1` es más poderosa que `M2`.
Notar que este predicado no consiste en contar los elementos en las mochilas o sumar sus potenciales, dado que el potencial depende de las reglas de composición asociadas a los símbolos `binaria` y `jerarquica`.

Ejercicio 4

Una mochila `M1` es mejor que otra `M2` cuando para cada configuración C_2 de `M2` se puede encontrar una configuración C_1 de `M1` que tiene tanto o más potencial que C_2 y cuyo costo es menor al de C_2. Dar un predicado `mejor(+M1,+M2)` que es verdadero cuando `M1` es mejor que `M2`.

Ejercicio 5

En algunas instancias del juego se deberá particionar a la mochila para componer distintas herramientas con distintos potenciales. Por ejemplo, se podrían necesitar usar los elementos

de la mochila para componer dos herramientas, una con un potencial mínimo de 30 y otra de 80. Se desea definir un predicado que a partir de una mochila `M` y una lista `Ps` de potenciales, genere una lista `Cs` de herramientas compuestas que satisfagan los potenciales solicitados.
Se solicita definir el predicado cuaternario `usar(+M1,+Ps,?Cs,?M2)`, donde `M1` es la mochila original, `Ps` es la lista de potenciales requeridos, `Cs` es una lista de herramientas compuestas y `M2` es el conjunto de elementos no utilizados. El predicado `usar(+M1,+Ps,?Cs,?M2)` es verdadero cuando `Ps` y `Cs` tienen la misma longitud y cada elemento de `Cs` tiene un potencial mayor o igual al elemento que se encuentra en la misma posición de `Ps`. Además, cada elemento de `M1` o bien se utiliza para construir una única composición de `Cs` o bien aparece como un elemento no usado en `M2`.
Por ejemplo, la consulta:
`usar([rayo,rayo,volatilizador,rayo,encendedor],[30,80],C,[encendedor]).`
que requiere conformar dos herramientas con potenciales `30` y `80` a partir de la mochila `[rayo,rayo,volatilizador,rayo,encendedor]` sin utilizar `encendedor`, tiene (entre otras) las siguientes soluciones

```
C = [binaria(rayo, rayo), jerarquica(rayo, volatilizador)]
C = [binaria(rayo, rayo), binaria(volatilizador, rayo)]
C = [jerarquica(rayo, rayo), jerarquica(rayo, volatilizador)]
```

Ejercicio 6

En ciertas ocasiones se deberá comprar una mochila `M` con una cantidad máxima `C` de herramientas básicas y que permita componer una herramienta con un determinado potencial. Definir un predicado ternario `comprar(+P,+C,?M)` que es verdadero cuando `M` tiene al máximo `C` herramientas básicas y permite construir una herramienta con potencial mayor o igual a `P`. Por ejemplo, la consulta `comprar(100,4,M)` retorna (entre otras) las siguientes soluciones:

```
M = [rayo,rayo]
M = [rayo, volatilizador]
M = [volatilizador, volatilizador]
M = [rayo, rayo, rayo]
```

2. Pautas de Entrega

El principal objetivo de este trabajo es evaluar el correcto uso del lenguaje PROLOG de forma declarativa para resolver el problema planteado.
Se debe entregar el código impreso con la implementación de los predicados pedidos. Cada predicado asociado a los ejercicios debe contar con ejemplos que muestren que exhibe la funcionalidad solicitada. Además, se debe enviar un e-mail conteniendo el código fuente en Prolog a la dirección plp-docentes@dc.uba.ar. Dicho mail debe cumplir con el siguiente formato:

- El título debe ser `[PLP;TP-PL]` seguido inmediatamente del nombre del grupo.
- El código Prolog debe acompañar el e-mail y lo debe hacer en forma de archivo adjunto con nombre `tp2.pl`.

El código debe poder ser ejecutado en SWI-Prolog. No es necesario entregar un informe sobre el trabajo, alcanza con que el código esté adecuadamente comentado. Los objetivos a evaluar en la implementación de los predicados son:

- corrección,
- declaratividad,
- reutilización de predicados previamente definidos
- utilización de unificación, backtracking, generate and test y reversibilidad de los predicados.
- **Importante**: salvo donde se indique lo contrario, los predicados no deben instanciar soluciones repetidas ni colgarse luego de devolver la última solución. Vale aclarar que no es necesario filtrar las soluciones repetidas si la repetición proviene de las características de la entrada.

Importante: se admitirá un único envío, sin excepción alguna. Por favor planifiquen el trabajo para llegar a tiempo con la entrega.

3. Referencias y sugerencias

Como referencia se recomienda la bibliografía incluída en el sitio de la materia (ver sección *Bibliografía* → *Programación Lógica*).
Se recomienda que utilicen los predicados ISO y los de SWI-Prolog ya disponibles, siempre que sea posible. Recomendamos especialmente examinar los predicados y metapredicados que figuran en la sección *Cosas útiles* de la página de la materia. Pueden hallar la descripción de los mismos en la ayuda de `SWI-Prolog` (a la que acceden con el predicado `help`). También se puede acceder a la documentación online de SWI-Prolog.