Laboratorio de Lenguajes de Programación

USB / CI-3661 / Ene-Mar 2022

(Programación Lógica – 35 puntos)

Iterative Deepening

Construya una infraestructura para resolver problemas de búsqueda aplicando la técnica de Iterative Deepening (IDS). El predicado principal será

```
iddfs(Problema,Estado,Movimientos)
```

cuyos argumentos tienen el mismo significado que `dfs/3` discutido en clase. Adapte los predicados `resolver/2` y `simular/1` para que usen la maquinaria Iterative Deepening.

La profundidad de búsqueda *debe ser inaccesible* para el usuario del predicado de alto nivel. Así mismo, no hay cota para la profundidad máxima de búsqueda, por lo tanto *siempre* debe ser posible aumentar la profundidad vía *backtracking*.

En conjunto, los tres predicados aportarán puntos si, y sólo si, son usdos efectivamente para resolver:

Patio de Operaciones (8 puntos)

Ud. es el operador de cambia vías del Yalda de Calacras – el casi-metro de una urbe distópica futurística. Las formaciones (o "trenes", para el pueblo) se especifican como una lista Prolog

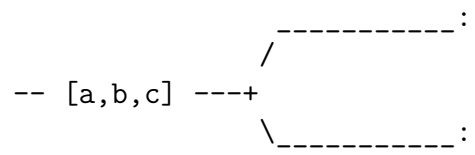
```
[c1,c2,c3,...,cN]
```

donde los `cI` son los vagones. La máquina principal está ubicada del lado de `c1` (como si fuera `c0`), pero no está indicada explícitamente porque sólo interesan los vagones.

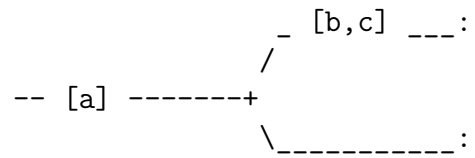
Ud. se encuentra en el patio de operaciones, junto a la "Y" de intercambio. La formación llega por el pie de la "Y" y puede usar los brazos de la "Y" para tomar o dejar vagones. Su trabajo es transformar la formación que llega a una permutación específica.

Por ejemplo, si llega la formación `[a,b,c]` y se necesita reordenarla para tener la formación `[b,c,a]`, podría procederse de la siguiente manera. (Imagine que la "Y" está acostada)

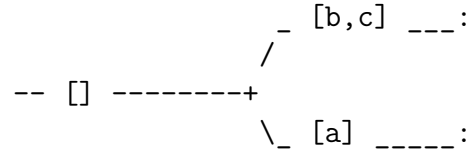
- Llegada del tren, por la base de la "Y".



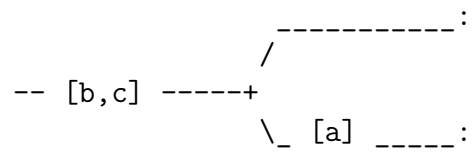
- Retrocede al brazo superior, desengancha los dos últimos vagones, `[b,c]`, y regresa a la base de la "Y".



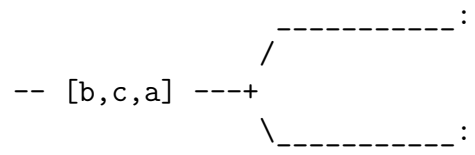
- Retrocede al brazo inferior, desengancha el único vagón, `[a]`, y regresa a la base de la "Y" – recuerde que la locomotora es tácita.



- Retrocede al brazo superior, engancha `[b,c]` y regresa a la base de la "Y".



- Retrocede al brazo inferior, engancha `[a]` y regresa a la base de la "Y".



Esta serie de operaciones son representadas por la siguiente secuencia de instrucciones

```
[push(above,2), push(below,1), pop(above,2), pop(below,1)]
```

Si el significado de la lista no es obvio, piense que `above` y `below` hacen referencia a los brazos superior e inferior de la "Y"; que `push` y `pop` hacen referencia a dejar o tomar vagones; y que el número es la cantidad de vagones involucrados en la operación.

Se desea que Ud. implante el predicado `vagones/3` que sea invocado de la forma

```
?- vagones([a,b,c],[b,c,a],Operaciones).
```

y triunfe unificando `Operaciones` con la lista resultante, que además debe ser la solución *más corta* posible para el problema.

Su predicado `vagones/3` **debe** apoyarse en la maquinaria IDDFS para encontrar la solución. Esto es, *debe* implantar `vagones/3` como una llamada al `iddfs/3` desarrollado en la pregunta anterior. Debe ser posible invocar `vagones/3` tantas veces cómo se desee en la misma sesión.

Alternativamente, puede usar la máquina DFS discutida en clase *verbatim*, combinada con predicados auxiliares, para resolver el problema. En este caso, sólo obtendrá cuatro (4) puntos como máximo.

Control Remoto (8 puntos)

Un adicto a la televisión tiene un televisor en cada una de las cuatro esquinas de su habitación. Están encendidos constantemente, de manera que no importa hacia donde mire el adicto, tenga la televisión a la vista.

Todos los televisores reciben `N` canales, numerados 1 a `N`. Al adicto no le importa lo que estén pasando. Lo único que le importa es que *todos* los televisores estén en el mismo canal al mismo tiempo.

Cada televisor tiene su propio control remoto, pero solamente le funciona el botón «Next» para cambiar al siguiente canal. Encima, los televisores son tan viejos, que sólo pueden cambiar un canal a la vez: si están en el canal `C`, al oprimir el botón «Next» pasarán al canal `C+1 mod N`, pero no se podrá volver a aplicar «Next» hasta que *otro* televisor haya sido cambiado de canal.

El adicto se queda dormido, y a la mañana siguiente encuentra que los gremlins han cambiado los canales de los televisores, de manera que no coinciden. Obviamente, se desespera porque necesita que todos los televisores estén en el mismo canal.

Escriba el predicado `canales/3` tal que:

- El primer argumento siempre está instanciado con una lista de cuatro elementos, indicando el canal sintonizado en cada televisor.
- El segundo argumento siempre está instanciado con el número de canales disponibles.
- El predicado triunfa unificando su tercer argumento con una lista. La lista describe el orden en que hay que aplicar «Next» a cada televisor, para que al final terminen sintonizados en el mismo canal. Más aún debe ser la secuencia *mínima* de pasos.

Por ejemplo,

```
?- canales([3,1,3,3],5,L).
L = [1,2,3,2,4,2]
yes
```

pues si los televisores comienzan sintonizados en `[3,1,3,3]`, después de aplicar «Next» al TV1 TV2, TV3, TV2, TV4 y TV2, todos terminan sintonizados en `[4,4,4,4]`.

Alternativamente, puede usar la máquina DFS discutida en clase *verbatim*, combinada con predicados auxiliares, para resolver el problema. En este caso, sólo obtendrá cuatro (4) puntos como máximo.

Aritmética Racional (11 puntos)

Implemente un conjunto de predicados para hacer aritmética racional. Los números racionales, o fracciones exactas, serán representado de la forma `n\\d` donde `n` es el numerador, y `d` el denominador. Su implantación debe incluir las operaciones:

- `A isr E` – tal que evalúa `E` e instancia `A` al resultado.
- `A + B` – suma de racionales
- `A - B` – diferencia de racionales
- `A * B` – producto de racionales
- `A / B` – cociente de racionales
- `~A` – recíproco (inverso) de racionales.

Los resultados de `isr/2` siempre deben presentarse en la forma reducida más simple posible, e.g.

```
?- consult('ratio.pro').
?- A isr 20\\14.
A = 10\\7

?- A isr 20\\14 + 4\\7.
A = 2

?- A isr ~2.
A = 1\\2

?- A isr ~2\\3 + 5.
A = 13\\2

?- A isr 0 - (2\\5 / 10\\3).
A = -3\\25

?- A isr 1\\0.
A = infinity

?- A isr 0\\0.
A = not_a_number
```

No se puede operar con `infinity` ni `not_a_number`, sólo se propaga su ocurrencia.

Su implementación debe tener la siguiente estructura general:

- Definición de los operadores necesarios, de manera que complementen la jerarquía estándar de operadores Prolog.
- Un predicado `simplify/2` que triunfa si su segundo argumento es el número racional más simple equivalente a su primer argumento.
- La implantación del predicado `isr/2`.

Tablas de la verdad

Escriba el predicado `truthtable/0` **(6 puntos)** tal que al invocarlo se emita un prompt en pantalla en el cual el usuario escriba una fórmula lógica arbitraria, e inmediatamanete se muestre en pantalla la Tabla de Verdad asociada.

```
?- truthtable.
formula: x :\/: y.
+---+---+---+
| x | y | F |
+---+---+---+
| t | t | t |
| t | f | t |
| f | t | t |
| f | f | f |
+---+---+---+
formula: x :\/: :~: y :/\: z.
+---+---+---+---+
| x | y | z | F |
+---+---+---+---+
| t | t | t | t |
| t | t | f | t |
| t | f | t | t |
| t | f | f | t |
| f | t | t | f |
| f | t | f | f |
| f | f | t | t |
| f | f | f | f |
+---+---+---+---+
formula: bye.
```

Las fórmulas pueden tener una cantidad arbitraria de variables, que pueden ocurrir más de una vez. Puede suponer que los argumentos de los conectores lógicos siempre estarán instanciados con átomos concretos de una letra. El único átomo especial será `bye` que permitirá terminar la invocación del predicado.

Los conectores lógicos deben tener la precedencia acostumbrada: la negación (`:~:`) precede a la conjunción (`:/\:`), que precede a la disyunción (`:\/:`), y los paréntesis pueden usarse libremente.

Las variables deben ser presentadas en la tabla en el mismo orden de ocurrencia que tienen en la fórmula original.

Sólo después de completar `truthtable/0`, implante el predicado `truthtablepro/0`, tal que los nombres de variables sean de más de una letra, y que en la tabla aparezca la fórmula concreta en lugar de `F`, con su resultado centrado, es decir

```
?- truthtablepro.
formula: esto :\/: lo_otro.
+------+---------+-----------------+
| esto | lo_otro | esto:\/:lo_otro |
+------+---------+-----------------+
|   t  |    t    |        t        |
|   t  |    f    |        t        |
|   f  |    t    |        t        |
|   f  |    f    |        f        |
+------+---------+-----------------+
```

para obtener **(2 puntos)**.

Detalles de la Entrega

Prepare su entrega en un archivo `pp-<g>.tgz`, donde `<g>` es su número de grupo oficial. El `tgz` debe contener los archivos con las soluciones.

- `busqueda.pro` – la maquinaria IDDFS o DFS, y los predicados que Ud. considere necesarios para resolver «Patio de Operaciones» y «Control Remoto».
- `racionales.pro`
- `tabla.pro`

Preste atención a que su proyecto funcione con `SWI-Prolog` 8.2 pues esa es la versión que usaré para evaluar.

Envíe ese archivo como adjunto a mi dirección de contacto a más tardar el Viernes 2022-04-29 a las 23:59 VET. Cada minuto de retraso en la entrega le restará un (1) punto de la calificación final.

–

Prof.Ernesto Hernández-Novich

<emhn@usb.ve>

2022-04-15