1. Enunciado

Introducción a la programación, guía 2, ejercicio 5: Frecuencia de bondis.

A Ciudad Universitaria (CU) llegan 8 lineas de colectivos (28, 33, 34, 37, 45, 107, 160,166). Con el fin de controlar la frecuencia diaria de cada una de ellas, un grupo de investigación del Departamento de Computación instaló cámaras y sistemas de reconocimiento de imágenes en el ingreso al predio. Durante cada día dicho sistema identifica y registra cada colectivo que entra, almacnando la información de a qué línea pertenece en una secuencia.

a) Especificar el problema cantidadColectivosLinea() que a patir de la secuencia de colectivos que entran a CU, el número de una de las líneas que entra a CU, y una secuencia que cumpla con la descrpción del sistema presentado, devuelva cuántos colectivos de esa línea ingresaron durante el día.

b) Especificar el problema compararLineas() que a partir de los números de 2 líneas y una secuencia que cumpla con la descripción del sistema presentado, devuelva cuál de las dos líneas tiene ayor frecuencia diaria (utilizar cantidadColectivosLinea())

2. Elección

El ejercicio elegido es un problema de especificación porque me parece un tema que los estudiantes suelen pasar por alto pero, como estudiante de datos, me ayudó a familiarizarme con el uso de operadores y conectores lógicos indispensables para la formalidad matemática de la carrera. Además es importante contar con una manera de expresar los problemas sin las ambigüedades del español.

Lo elegí de la nueva guía porque serán esos los nuevos problemas a enseñar de este tema.

En esta guía encontré solo dos ejercicios vinculados a realizar una especificación desde 0. Se puede repasar en clase lo visto en la teórica correspondiente antes de comenzar. Me parece un buen ejercicio para entrar en calor al recién estar introduciéndose a los conceptos.

Además me parece interesante mostrar la ventaja de modularizar. Aquí vamos a poder utilizar el mismo predicado para distintas circunstancias y el hecho de tenerlo así hace que no debamos escribir muchas veces lo mismo.

3. Preconceptos

Se asume que el alumno ya maneja los significados correspondientes a cada conector lógico utilizado (diferencias entre implica/ implica Luego/ y / y Luego / o / etc). Obviamente siempre se pueden repasar brevemente en clase. Además deben tener nociones teóricas básicas de qué significa espeficicar (básicamente haber ido a la teórica).

4. Resolución ítem a

4.1. Parámetros

En una especificación, los parámetros son los datos de entrada (con los que contamos) y de salida (lo que necesitamos) con los que estamos trabajando. En este caso, el enunciado nos dice que contamos con la secuencia de líneas de colectivos de CU (secuencia de enteros), el número de una línea de colectivo a analizar (un entero) y una secuencia que fue registrando la información de qué colectivo iba entrando a CU (otra secuencia de enteros). Necesitamos saber cuál colectivo fue el que más entró a CU (lo podemos representar con un entero). Podemos ir pensando que esto vendrá dado por el colectivo que más aparece en el registro de bondis que entraron a CU.

Recordemos que los parámetros de entrada se escriben con un IN adelante (no hay ninguno INOUT ya que nuestro resultado es obtenido a partir de los parámetros pero no se corresponde con uno solo de los originales). Al parámetro de salida se lo escribe luego de los de entrada y es uno solo: un entero que representa la cantidad de veces que el colectivo n entró a CU.

```
proc cantidadColectivosLinea (in c: seq(ℤ), in n: ℤ, in s: seq( ℤ) : ℤ)  {
}
```

4.2. Precondición

Pensemos a la precondición como un contrato entre el problema y el usuario. Por ejemplo, si yo tengo un programa que me dice si va a llover en Buenos Aires, no le puedo pedir que me diga si va a llover en Los Ángeles.

Si yo tengo un registro de los colectivos que entran a CU, no puedo pedirle información acerca de los colectivos que entran a Constitución.

Una precondición de un problema de especificación en particular está fuertemente asociada con asegurarnos que los datos dados son válidos en el contexto de nuestro problema. Analicemos parámetro por parámetro entendiendo qué es lo que representa cada uno y cómo plasmar la validez de cada uno.

Tengamos en cuenta que no sirve que un solo parámetro sea válido, necesitamos que todos lo sean.

Parámetro	¿Qué representa?	Validez (más informal)	Validez (menos informal)
c	Líneas de colectivos que entran a CU (secuencia de enteros).	Todos los elementos de c son colectivos que entran a CU (todos los elementos de c pertenecen a la secuencia de los 8 posibles). Son EXACTAMENTE los 8 posibles. No hay repetidos	Para todo elemento de c, ese elemento pertenece a los 8 posibles. Para todo elemento de los 8 posibles, ese elemento pertenece a c. No existen dos índices distintos de c con el mismo elemento.
n	Colectivo a analizar (entero)	Tiene que ser un colectivo que entra a CU (n pertenece a c).	Existe un elemento en c tal que n=elementoDeC
s	Registro del sistema de colectivos de CU durante el día (secuencia de enteros)	Todos los elementos de s tienen que ser colectivos que entran a CU (es decir, todos los elementos de s pertenecen a c).	Para todo elemento de s, ese elemento pertenece a c

Para mayor legibilidad se pueden hacer predicados donde plasmaremos en lenguaje formal lo que queremos expresar.

Es razonable ir pensando que el predicado de pertenencia lo vamos a usar varias veces.

Además tengamos en cuenta que cuando hablamos de que 'todos los elementos de una secuencia pertenecen a otra' estamos hablando de que 'para cada elemento, ese pertenece a la otra'- Por lo tanto, el predicado de pertenencia de secuencias va a utilizar al de pertenencia de un solo elemento. Esto lo podemos hacer porque los elementos 'tipan'.

Es decir, cuando estamos viendo cada elemento de una secuencia de enteros, estamos viendo a un entero y miramos la pertenencia con un predicado que involucra a un entero y a otra secuencia de enteros.

Si nuestra primera secuencia hubiera sido una secuencia de strings no podríamos reutilizar el mismo predicado de pertenencia que mira un entero en una secuencia de enteros.

4.3. Postcondición

La postcondición es lo que nuestro problema quiere asegurar. En este caso queremos que nuestro resultado sea la cantidad de veces que nuestro colectivo n entró a CU según el registro de s. Es decir, la cantidad de veces que n aparece en s.

Podríamos garantizar esto mediante una sumatoria entre todos los elementos de s, es decir entre todos los $s[i]$ (donde i es un índice de la secuencia). Si el elemento que estamos viendo es nuestro n que sume 1 y si no, 0.

Luego, la postcondición diría que el resultado es igual a la sumatoria de recién.

4.4. Juntando todo

Los parámetros ya sabemos cómo se escriben, la precondición es la que asegura la validez de nuestros datos (ya sabemos qué vendría a ser eso) y nuestra postcondición ¡también la tenemos! Estamos en condiciones de empezar a especificar más formalmente.

proc `cantidadColectivosLinea` (in c seq($\mathbb{Z}$), in n $\mathbb{Z}$, in s: seq($\mathbb{Z}$) : $\mathbb{Z}$) {

Pre $\equiv$ {todosPertenecen(c, [28,33,34,37,45,107,160,166]) $\wedge$ todosPertenecen([28,33,34,37,45,107,160,166],c) $\wedge$ sinRepetidos(c) $\wedge$ pertenece(n,c) $\wedge$ todosPertenecen(s,c)}

Post $\equiv$ {resultado = $\sum_{i=0}^{|s|-1}$(if (s[i]=n) then 1 else 0)}

}

Sabemos por la tabla de más arriba qué significa cada una de las precondiciones. Utilicemos la tabla para escribir cada uno de esos predicados que ya definimos con un lenguaje un poco más lógico todavía.

proc `cantidadColectivosLinea` (in c seq($\mathbb{Z}$), in n $\mathbb{Z}$, in s: seq($\mathbb{Z}$) : $\mathbb{Z}$) {

Pre $\equiv$ {todosPertenecen(c, [28,33,34,37,45,107,160,166]) $\wedge$ todosPertenecen([28,33,34,37,45,107,160,166],c) $\wedge$ sinRepetidos(c) $\wedge$ pertenece(n,c) $\wedge$ todosPertenecen(s,c)}

Post $\equiv$ {resultado = $\sum_{i=0}^{|s|-1}$(if (s[i]=n) then 1 else 0)}

pred `todosPertenecen` (a: seq($\mathbb{Z}$), b: seq($\mathbb{Z}$)) {

$(\forall$ i $\in \mathbb{Z})(0 \leq i < |a| \longrightarrow_L pertenece(a[i], b))$

}

pred `sinRepetidos` (a: seq($\mathbb{Z}$)) {

$(\forall$ i,j $\in \mathbb{Z})(0 \leq i < j < |a| \longrightarrow_L a[i] \neq a[j])$

}
pred pertenece (n: $\mathbb{Z}$, a: seq($\mathbb{Z}$)) {
$(\exists\ \mathrm{i} \in \mathbb{Z})(\ 0 \leq i < |a| \wedge_L a[i] = n)$
}
}

Es importante recordar que en los predicados, cuando estamos utilizando elementos indexados en la secuencia, utilizamos 'implica Luego' o 'y Luego' para salvar indefiniciones.

5. Resolución ítem b

Ahora contamos con 2 enteros. Llamémoslos $c1$ y $c2$ que son 2 líneas de colectivos y una secuencia s con las frecuencias de entrada. Notemos la similitud del $c1$ y el $c2$ con el n del ítem a y de la secuencia s con la secuencia s del ítem a.

Luego, las precondiciones van a ser casi las mismas que en el problema anterior, con la diferencia que asumimos a [28,33,34,37,45,107,160,166] como válida (nos ahorramos las precondiciones que en el item a provenian de ver que c fuese válida).

Para $c1$ y $c2$ las precondiciones serán las misma que usamos para el n en el item a.

Para s es la misma que usamos para la s del item a.

Lo que queremos ver en la post es que si nuestro resultado es el colectivo $c1$ es porque la cantidad de veces que aparece $c2$ en s es menor (o a lo sumo igual) que la que aparece $c1$.

Análogamente, si nuestro resultado es el colectivo $c2$ es porque la cantidad de veces que aparece $c1$ en s es menor (o a lo sumo igual) que la que aparece $c2$.

Usemos que ya teníamos un problema que miraba las cantidades (en la antigua Algo 1 sería una función auxiliar distinta ya que no se usaban procs en otros procs).

Los preds definidos en el item a para la pertenencia no hace falta volverlos a escrbir, bastaría con hacerlos predicados globales auxiliares.

Como en el enunciado dice que usemos el problema anterior, explicitamos el parámetro de las líneas de colectivo a chequear al ver que n y s sean válidos porque no tenemos a c como parámetro y hacemos lo mismo a la hora de llamar al problema anterior.

proc compararLineas (in c1: $\mathbb{Z}$, in c2 $\mathbb{Z}$, in s: seq($\mathbb{Z}$) : $\mathbb{Z}$) {
Pre $\equiv$ {pertenece(c1,[28,33,34,37,45,107,160,166]) $\wedge$ pertenece(c2,[28,33,34,37,45,107,160,166]) $\wedge$
todosPertenecen(s,[28,33,34,37,45,107,160,166])}
Post $\equiv$ {
(resultado = c2 $\wedge$*cantidadColectivosLinea*([28, 33, 34, 37, 45, 107, 160, 166], $c1, s$) $\leq$
cantidadColectivosLinea([28,33,34,37,45,107,160,166],c2,s))
$\vee$
(resultado = c1 $\wedge$*cantidadColectivosLinea*([28, 33, 34, 37, 45, 107, 160, 166], $c2, s$) $\leq$
cantidadColectivosLinea([28,33,34,37,45,107,160,166],c1,s)) }
}