Tecnología Digital 1: Introducción a la Programación – Recuperatorio del Trabajo Práctico 1

Autores: Sosa, Luisina
Condorpocco, Vanina
Texeira, Milagros

Función simBP

Terminación del ciclo:
Para demostrar que termina, debemos entender cuándo termina el ciclo, según la guarda. En este caso:

“i<longitud_binario_n and i<longitud_binario_m”

Por lo tanto, nuestro ciclo dejará de ejecutarse en tanto nuestra variable **i** sea igual o más grande que **longitud_binario_n** o **longitud_binario_m**, esto por lo que nos dice la regla del and que si una de las condiciones es false toda la condición conjunta es false. Para ver que esto sucede, podemos seguir la siguiente línea de pensamiento:

- Nuestra variable **i** se inicializa en 0.
- Como en el *Requiere* nos especifican que **n>0** y **m>0**, **longitud_binario_n** y **longitud_binario_m** siempre serán mayores a 0, por lo tanto el ciclo siempre se va a ejecutar al menos una vez. Veamos qué pasa al ejecutarse el ciclo.
- La variable **i** se va incrementando en 1 cada vez que el ciclo se corre. Además, hay un caso donde **i** vale más que **longitud_binario_n** y es cuando no se cumple la condición **if(num_binario_n[i]==num_binario_m[i])** esto hace que **i** valga **longitud_binario_n + 1** al llegar al final del ciclo. Cuando esto suceda, **i** no es menor a **longitud_binario_n** por lo que la condición para que se ejecute el ciclo (**i<longitud_binario_n and i<longitud_binario_m**) será false, ocasionando que este termine. Veamos qué pasa si esa condición **if** siempre se cumple.
- Las variables **longitud_binario_n** y **longitud_binario_m** no se modifican mientras el ciclo se ejecuta, por lo que son constantes.
- Por lo tanto, es inevitable que en alguna iteración, i alcance el valor de **longitud_binario_n** o **longitud_binario_m.**
- Cuando esto sucede, **i** no es menor a **longitud_binario_n o longitud_binario_m**, por lo que la condición (**i<longitud_binario_n and i<longitud_binario_m**) cuando se ejecute el ciclo será false, ocasionando que este termine.

Predicado Invariante
Para cualquier valor de n y m (siguiendo la especificación):
***Inv*:** res vale la longitud del prefijo común más largo que hay en los primeros i caracteres entre las representaciones binarias de n (num_binario_n) y m (num_binario_m).

Correctitud:
Este predicado, vale para cada iteración del ciclo, por lo que podemos sacar la siguiente conclusión:

El ciclo termina cuando i >= longitud_binario_n o si i>= longitud_binario_m.

Por lo tanto, podemos reemplazar i por longitud_binario_n o longitud_binario_m en nuestro invariante.

Si hacemos esto, nuestro invariante nos asegura que “res vale la longitud del prefijo común más largo que hay en los primeros longitud_binario_n o longitud_binario_m caracteres entre las representaciones binarias de n (num_binario_n) y m (num_binario_m)”.

Pero, decir la longitud del prefijo común más largo que hay en los primeros longitud_binario_n o longitud_binario_m caracteres entre las representaciones binarias de n (num_binario_n) y m (num_binario_m) es lo mismo que decir, la similitud binaria de prefijo entre n y m.

Entonces: al terminar el ciclo, “res vale la similitud binaria del prefijo entre n y m”.
Y esa es la conclusión a la que queríamos llegar.

Función lista_elementos_simBP_con_N

Terminación del ciclo:
Para demostrar que termina, debemos entender cuando termina el ciclo, según la guarda.
En este caso:

“i<longitud_xs”

Por lo tanto, nuestro ciclo dejará de ejecutarse en tanto nuestra variable i sea igual o mas grande que longitud_xs. Para ver que esto sucede, podemos seguir la siguiente línea de pensamiento:

- Nuestra variable i se inicializa en 0, antes de que arranque el ciclo.
- Nuestra variable longitud_xs es la cantidad de elementos que tiene la lista xs, lo cual es si o si >=0. Si es 0 el ciclo no se ejecuta nunca y como vr es inicializado en 0, devolveremos 0. Veamos qué pasa si longitud_xs>0.
- Hay que hacer una mención especial a que dentro del ciclo usamos una función auxiliar llamada simBP() que contiene un ciclo, aclarado, continuamos.
- La variable i se va incrementando en 1 cada vez que el ciclo se corre.
- La variable longitud_xs no se modifica mientras el ciclo se ejecuta, por lo que es una constante.
- Por lo tanto, es inevitable que en alguna iteración, i alcance el valor de longitud_xs.
- Cuando esto suceda, i<longitud_xs sera falso y el ciclo terminará.

Predicado invariante:
Inv**:** vr es una lista que contiene, en el mismo orden en que aparecen en xs, las simBPs entre n y cada uno de los primeros i elementos de xs

Correctitud:
El ciclo termina cuando i==longitud_xs.
Entonces reemplazamos i por longitud_xs en inv y sabemos que al terminar el ciclo, "**vr** es una lista que contiene, en el mismo orden en que aparecen en xs, las simBPs entre n y cada uno de los primeros longitud_xs elementos de xs".
Decir "los primeros longitud_xs elementos de s" equivale a decir "todos los elementos de s".
Entonces, al terminar el ciclo, vr es una lista que contiene, en el mismo orden en que aparecen en xs, las simBPs entre n y cada uno de los elementos (todos) de xs.
Eso es exactamente lo que se espera que devuelva la función según su especificación.