Prueba de Oposición

Eric Brandwein

5 de Mayo de 2023

Algoritmos y Estructuras de Datos I

1. Lógica
2. Introducción al Lenguaje de Especificación
3. **Especificación de problemas**
4. Precondición más débil en SmallLang
5. Demostración de corrección de ciclos en SmallLang
6. Testing
7. Ciclos a partir de invariantes
8. Tiempo de Ejecución de Peor Caso de un Programa

Guía 3: Especificación de Problemas

Con esta guía se busca que el estudiante obtenga las herramientas básicas para después poder demostrar que un algoritmo es correcto. Se trata principalmente sobre el **contrato** de un algoritmo.

Guía 3: Especificación de Problemas

Con esta guía se busca que el estudiante obtenga las herramientas básicas para después poder demostrar que un algoritmo es correcto. Se trata principalmente sobre el **contrato** de un algoritmo.

- Se introducen los conceptos de pre- y poscondición.

Guía 3: Especificación de Problemas

Con esta guía se busca que el estudiante obtenga las herramientas básicas para después poder demostrar que un algoritmo es correcto. Se trata principalmente sobre el **contrato** de un algoritmo.

- Se introducen los conceptos de pre- y poscondición.
- Se comienzan a implementar especificaciones de problemas con los `proc`.

Ejercicio

El siguiente ejercicio nos pide analizar un `proc` y determinar si es correcto o no, para familiarizarnos con el concepto de pre- y poscondición antes de comenzar a implementarlos nosotros mismos.

Ejercicio 2

Veamos la siguiente especificación de un problema:

```
proc elementosQueSumen (
```
in $l : seq\langle\mathbb{Z}\rangle$, in $suma : \mathbb{Z}$, out $result : seq\langle\mathbb{Z}\rangle$
```
) {
```
Pre $\{True\}$

Post $\{contenida(result, l) \land suma = \sum_{i=0}^{|result|-1} result[i]\}$
```
}
```

```
pred contenida (
```
$adentro : seq\langle\mathbb{Z}\rangle, afuera : seq\langle\mathbb{Z}\rangle$) {

$(\forall x : \mathbb{Z})(\#apariciones(x, adentro) \leq \#apariciones(x, afuera))$
```
}
```

Ejercicio 2

Veamos la siguiente especificación de un problema:

```
proc elementosQueSumen (
    in l : seq⟨ℤ⟩, in suma : ℤ, out result : seq⟨ℤ⟩
) {
    Pre {True}
    Post {contenida(result, l) ∧ suma = ∑_{i=0}^{|result|−1} result[i]}
}

pred contenida (adentro : seq⟨ℤ⟩, afuera : seq⟨ℤ⟩) {
    (∀x : ℤ)(#apariciones(x, adentro) ≤ #apariciones(x, afuera))
}
```

a) ¿Es válida esta especificación? En caso contrario, mostrar por qué.

Ejercicio 2

Veamos la siguiente especificación de un problema:

```
proc elementosQueSumen (
    in l : seq⟨ℤ⟩, in suma : ℤ, out result : seq⟨ℤ⟩
) {
    Pre {True}
    Post {contenida(result, l) ∧ suma = ∑_{i=0}^{|result|−1} result[i]}
}

pred contenida (adentro : seq⟨ℤ⟩, afuera : seq⟨ℤ⟩) {
    (∀x : ℤ)(#apariciones(x, adentro) ≤ #apariciones(x, afuera))
}
```

a) ¿Es válida esta especificación? En caso contrario, mostrar por qué.

b) ¿Qué tenemos que cambiar para que sea válida?

¡Gracias!