Tipos complejos

seq< T >: secuencia de tipo T

crear	$\langle\rangle$, $\langle x, y, z\rangle$
tamaño	$\|s\|$, length(s)
pertenece	$i \in s$
ver posición	$s[i]$
cabeza	head(s)
cola	tail(s)
concatenar	concat(s1, s2), $s1 + s2$
subsecuencia	subseq(s, i, j), $s[i..j]$
setear posición	setAt(s, i, val)

conj< T >: conjunto de tipo T

crear	$\{\}$, $\{x, y, z\}$
tamaño	$\|c\|$, length(c)
pertenece	$i \in c$
unión	$c1 \cup c2$
intersección	$c1 \cap c2$
diferencia	$c1 - c2$

dict<K, V>: diccionario que asocia claves de tipo K con valores de tipo V

crear	$\{\}$, {"juan" : 20, "diego" : 10}
tamaño	$\|d\|$, length(d)
pertenece (hay clave)	$k \in d$
valor	d[k]
setear valor	setKey(d, k, v)
eliminar valor	delKey(d, k)

tupla<T1, ..., Tn>: tupla de tipos T1, ..., Tn

crear	$\langle x, y, z\rangle$
campo	$s[i]$

struct<campo1: T1, ..., campon: Tn>: tupla con nombres para los campos.

crear	$\langle x : 20, y : 10\rangle$
campo	s_x, s_y

Precondición más débil (wp)

La *precondición más débil* es la condición mínima necesaria antes de ejecutar una sentencia para garantizar que una postcondición dada se cumpla.

Lógica de Hoare

Una forma de razonar sobre la corrección de programas es a través de las triples de Hoare $\{P\}C\{Q\}$. El objetivo es obtener una fórmula lógica α tal que α es verdadera si y solo si $\{P\}C\{Q\}$ es verdadero.

Predicado def(E): Definición. Dada una expresión E, llamamos def(E) a las condiciones necesarias para que E esté definida.

Predicado Q_E^x: Definición. Dado un predicado Q, el predicado Q_E^x se obtiene reemplazando...

Axiomas WP

- **Axioma 1.** $\text{wp}(x := E, Q) \equiv \text{def}(E) \wedge Q_E^x$
- **Axioma 2.** $\text{wp}(\text{skip}, Q) \equiv Q$.
- **Axioma 3.** $\text{wp}(S1; S2, Q) \equiv \text{wp}(S1, \text{wp}(S2, Q))$.
- **Axioma 4.** Si $S =$ if B then $S1$ else $S2$ endif, entonces

$$\text{wp}(S, Q) \equiv \text{def}(B) \wedge_L ((B \wedge \text{wp}(S1, Q)) \vee (\neg B \wedge \text{wp}(S2, Q)))$$

- **Axioma 5.** $\text{wp}(\text{while } B \text{ do } S \text{ endwhile}, Q) \equiv (\exists_{i \geq 0})(H_i(Q))$ no podemos usar mecanicamente el Axioma 5 para demostrar la corrección de un ciclo con una cantidad no acotada a priori de iteraciones

Propiedades:

- Monotonía: Si Q implica R, entonces $\text{wp}(S, Q)$ implica $\text{wp}(S, R)$.
- Distributividad: $\text{wp}(S, Q \wedge R)$ equivale a $\text{wp}(S, Q) \wedge \text{wp}(S, R)$.
- Excluded Miracle: $\text{wp}(S, \text{false})$ equivale a false.

Invariante de un ciclo

Definición. Un predicado I es un invariante de un ciclo si:

1. I vale antes de comenzar el ciclo, y
2. Si vale $I \wedge B$ al comenzar una iteración arbitraria, entonces sigue valiendo I al finalizar la ejecución del cuerpo del ciclo.

Un invariante describe un estado que se satisface cada vez que comienza la ejecución del cuerpo de un ciclo y también se cumple cuando la ejecución del cuerpo del ciclo concluye.

Teorema del Invariante. Si existe un predicado I tal que:

1. $P_C \Rightarrow I$,
2. $\{I \wedge B\}S\{I\} \iff (I \wedge B) \implies wp(S, I)$
3. $I \wedge \neg B \Rightarrow Q_C$,

entonces $\{P_C\}$ while B do S endwhile $\{Q_C\}$ es válida.

Función variante

La función variante representa una cantidad que se va reduciendo

- Si existe una función variante f_v tal que:
  1. $\{I \wedge B \wedge f_v = v_0\}S\{f_v < v_0\} \iff (I \wedge B \wedge f_v = v_0) \implies wp(S, f_v < v_0)$
  2. Si $I \wedge f_v \leq 0$ implica $\neg B$

Correctitud de un Programa Completo

Para demostrar la correctitud de un programa completo utilizando la lógica de Hoare, seguimos estos pasos:

1. **Código antes del ciclo:** Debemos demostrar que las precondiciones implican la precondición más débil del código previo al ciclo, es decir,
$$Pre \Rightarrow \text{wp}(Código\_previo, P_c)$$
2. **Correctitud del ciclo:** Utilizamos el teorema del invariante, ya que no podemos calcular la precondición más débil del ciclo en general. El teorema del invariante se expresa de la siguiente manera:

   a)
   $$P_c \Rightarrow I$$
   b)
   $$(I \wedge B) \Rightarrow \text{wp}(Ciclo, I)$$
   c)
   $$(I \wedge \neg B) \Rightarrow Q_c$$
3. **Código posterior al ciclo:** Comprobamos que las postcondiciones del ciclo implican la precondición más débil del código posterior al ciclo, es decir,
$$Q_c \Rightarrow \text{wp}(Código\_posterior, Post)$$
4. **Conclusión:** Si probamos estas tres cosas, podemos concluir, por corolario de monotonía, que el programa completo es correcto con respecto a la especificación:
$$Pre \Rightarrow \text{wp}(Programa\_completo, Post)$$

Especificación y Relaciones de Fuerza

Especificación:
La especificación define qué es lo que debe hacer un algoritmo, en términos de relación entre sus entradas y sus salidas, sin determinar cómo lo hace. Las relaciones de fuerza entre especificaciones son importantes para entender cuán restrictiva o general es una especificación con respecto a otra.
Subespecificación:
Implica otorgar una precondición más restrictiva o una postcondición más débil que lo deducido del enunciado del problema. Una precondición más restrictiva excluye casos de entrada posibles. Una postcondición más débil permite soluciones no deseadas.
Sobreespecificación:
Consiste en proporcionar una postcondición más restrictiva o una precondición más débil que lo necesario. Una precondición más débil obliga al algoritmo a considerar casos innecesarios. Una postcondición más restrictiva limita las posibles soluciones.
Relaciones de Fuerza:
Decimos que A es más fuerte que B cuando $A \to B$ es una tautología. También podemos afirmar que A fuerza a B o que B es más débil que A.
Ejemplos:

- $p \wedge q$ es más fuerte que p.
- $p \vee q$ no es más fuerte que p.
- p es más fuerte que $p \to q$.
- p no es más fuerte que q.
- **False** es la fórmula más fuerte de todas.
- **True** es la fórmula más débil de todas.