MÉTODOS FORMALES · VERIFICACIÓN DE SOFTWARE

Correctitud Total en el Desarrollo Formal de Software

Julian Andres Lizalda Aristizabal
Luis Daniel Casierra Cuero
Juan David Riascos Garcia



La correctitud total exige dos garantías

Propiedad formal que asegura que un programa produce resultados correctos y siempre finaliza para cualquier entrada válida.

$$\text{Correctitud Total} = \text{Corrección Parcial} + \text{Terminación}$$

FÓRMULA CLAVE

Condición indispensable

No basta con entregar un resultado válido cuando la rutina frena: el sistema debe garantizar matemáticamente que nunca quedará atrapado en un estado divergente.

IMPORTANCIA CRÍTICA

Sistemas de alta integridad

Importancia: vital en software aeroespacial, marcapasos y sistemas financieros. Un bucle infinito puede ser catastrófico.

DIVERGENCIA

Lógica Formal

CONTRASTE

Corrección parcial no significa terminación

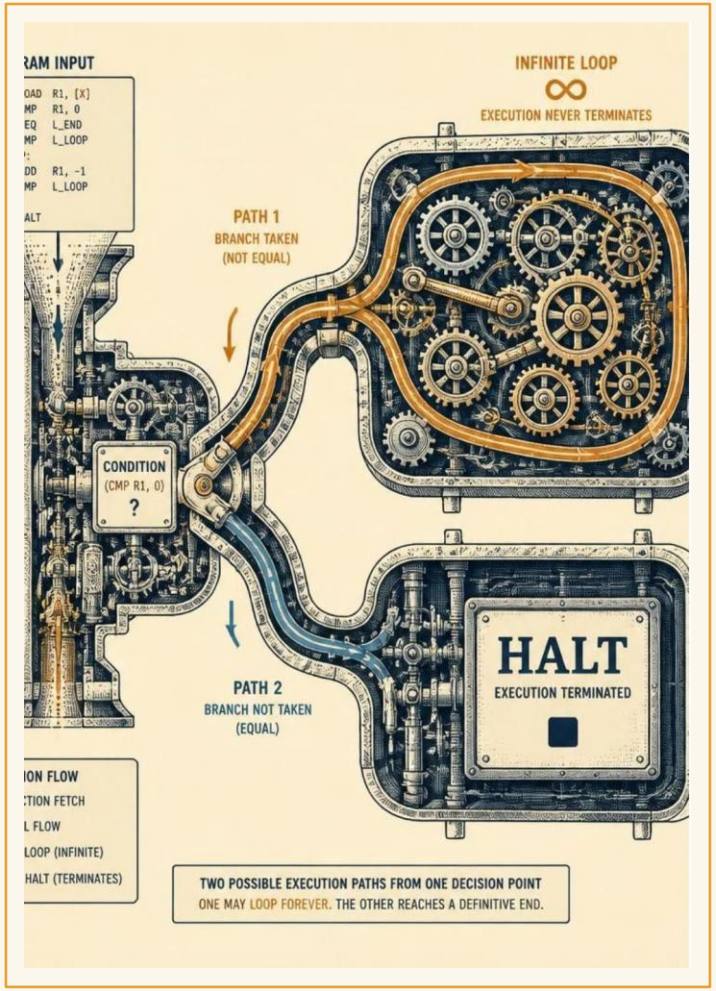
Corrección parcial: {P} S {Q}

Si P es verdadera y S termina, entonces Q será verdadera. No garantiza que S termine.

Corrección total: [P] S [Q]

Si P es verdadera, S termina garantizadamente y Q es verdadera al finalizar.

Integrante 2: demostración de la corrección parcial y papel de los invariantes.



Integrante 2

SEMÁNTICA

Axiomática

TRIPLETAS

La lógica de Hoare convierte el código en una prueba

Las tripletas de Hoare tienen la forma P {S} Q.

01

P, precondición

Estado inicial que debe cumplirse antes del código.

02

S, sentencia

Bloque de código que se verifica.

03

Q, postcondición

Estado que debe cumplirse al finalizar.

Razonamiento composicional: encadena afirmaciones lógicas para verificar programas secuenciales completos.

ITERACIÓN

Invariantes

PROPIEDADES

Los invariantes mantienen la propiedad a través del ciclo

Los ciclos while y for son difíciles de verificar porque su número de iteraciones puede variar.

1 **Inicialización**

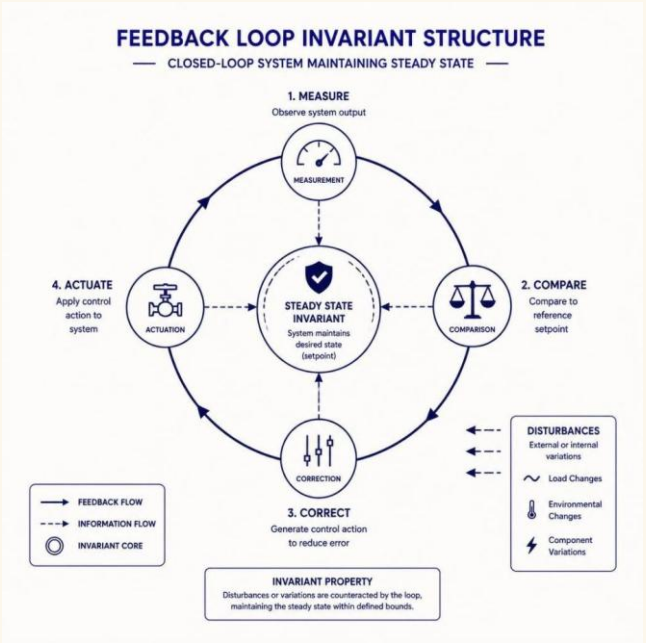
El invariante I es verdadero antes de entrar al ciclo.

2 **Mantenimiento**

I se mantiene verdadero después de cada iteración.

3 **Terminación lógica**

Al salir, I combinada con la condición de salida garantiza Q.



Integrante 3: demostración de la terminación y conclusiones.

Integrante 3

Una función variante demuestra que el ciclo no puede continuar para siempre

Para un ciclo o una función recursiva se usa una función variante v, también llamada cota de terminación.

v_inicio

v_k > v_{k+1} (descenso estricto)

Cota inferior (v = 0)

CONJUNTO ORDENADO

Vive en un conjunto bien ordenado, como los enteros no negativos.

DESCENSO ESTRICTO

Decrece estrictamente en cada iteración: v_nuevo < v_anterior.

COTA INFERIOR

Su valor mínimo marca la condición de salida.

PRÁCTICA

Ejemplo

CASO N

Ejemplo: decrementar n garantiza la terminación

```
while (n > 0) { n = n - 1; }
```

1

Precondición

Precondición: n ≥ 0, con n entero.

2

Función variante

Función variante: v = n.

3

Descenso

En cada paso, n disminuye en 1.

4

Terminación

Como n está acotado inferiormente por 0, el ciclo termina en a lo sumo n pasos.

Cálculo deductivo: el número de iteraciones está acotado superiormente por el valor inicial de n.

Verificación

SÍNTESIS

Conclusiones

CIERRE

La correctitud total une lógica y terminación

- ✓ **Garantía absoluta:** trasciende el testing mediante una demostración matemática.
- ✓ **Costo vs. beneficio:** exige esfuerzo analítico, pero es indispensable cuando fallar no es una opción.
- ✓ **Complementariedad:** los invariantes aseguran la lógica parcial; las funciones variante aseguran la ejecución.

