Resumen Analítico de Verificación Formal

Para que un problema encuentre solución es necesario como primer paso que se hayan escrito en forma clara y precisa los requerimientos de salida y los datos necesarios para obtener esa salida, es decir se debe *especificar el problema.*
La solución se expresa a través de un algoritmo que se convertirá en programa.
Es importante saber si éste está correctamente escrito, para lo cual se realiza un proceso de validación.
La validación permite comprobar que un *programa cumple con sus especificaciones*, es decir resuelve correctamente el problema para el que fue diseñado.
Los métodos de validación se pueden clasificar en dos grupos:

- Validación mediante pruebas (testing, se realiza ejecutando el programa)
- Validación mediante Verificación o Validación formal (se realiza sin ejecutar el programa)

Verificación
La Verificación consiste en *demostrar que el programa construido es correcto* respecto de la especificación dada**.**
La V*erificación Formal de Programas* consiste en un conjunto de técnicas de comprobación formales que permiten demostrar si un programa funciona correctamente.
Para interpretar el proceso de verificación hay ciertos conceptos que deben destacarse.
En la secuencia de acciones de un programa las variables que intervienen van tomando diferentes valores, pero estos valores no pueden ser cualquiera, deben respetar ciertas condiciones (por ejemplo, ser positivos, mayores a cero, etc.); esta situación hace que se determine un **Espacio de Estados**:

El **estado inicial** es el estado anterior a la ejecución de un código o parte de un código.
El **estado intermedio** viene determinado por los valores de las variables en cada momento.
El **estado final** es el estado posterior a la ejecución de dicho código.

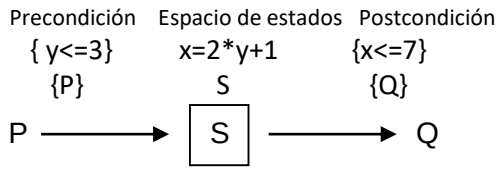
Hay condiciones llamadas **aserciones o asertos**, estas son expresiones lógicas que hacen referencia a un estado del programa y que se intercalan entre 2 acciones.
Una **aserción** es un predicado que se cumple (es verdadero) en determinado punto del programa.
Para indicar que una expresión es un aserto se encierra entre llaves, por ejemplo {P}.
Se dice que un **programa funciona correctamente** si cumple con especificaciones dadas. Para **especificar** un programa se utilizan los asertos que se cumplen al inicio y final del mismo, conocidos como precondición y postcondición respectivamente.

¿Qué elementos intervienen en una especificación formal?

- Un **Espacio de Estados** (universo de valores que pueden tomar las variables).
- **Precondición** (indican las condiciones que deben satisfacer los datos de entrada para que el programa pueda cumplir su tarea).
- **Postcondición** (indican las condiciones de salida que son aceptables como soluciones correctas del problema en cuestión).

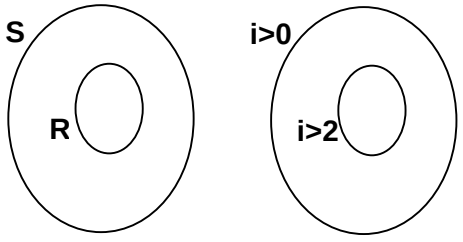
La precondición como la postcondición se especifican mediante predicados lógicos.

Podemos simbolizar el proceso realizado de la siguiente manera:



Implicación de predicado

Predicado más fuerte {R} y Predicado más débil {S}
Un predicado es más fuerte cuando es subconjunto de otro.



{i>2}	⇒	{ i >0}
Predicado más fuerte	Implicación	Predicado más débil

Aplicación de la lógica de Hoare a la verificación formal de Programas

Para la verificación de programas se utiliza un sistema de reglas basado en la **tripla de Hoare,** que permite especificar los programas basándose en lógica de predicados de primer orden.

Las especificaciones de un programa, **precondiciones y postcondiciones,** en la lógica de Hoare, son fórmulas denominadas **aserciones o asertos**. Para ello se escriben una terna del siguiente modo:

{P} S {Q} siendo P y Q aserciones lógicas

{Precondición} Programa {Postcondición}

Esta expresión se interpreta, si la precondición P es cierta antes de la ejecución del programa y dicho programa termina, entonces la postcondición Q es cierta tras la ejecución de dicho programa. Es decir que si se garantiza que la entrada actual satisface las restricciones de entrada (precondiciones) la salida satisface las restricciones de salida (postcondiciones).

Dicho de otro modo, los datos de entrada deben cumplir ciertas condiciones - *precondición* - (por ejemplo, que sean enteros y mayores que cero) para que se pueda ejecutar el programa, luego se ejecuta el *Programa* y se obtienen resultados, que son los datos de salida y también deben cumplir condiciones – *postcondición* - (por ejemplo, que una edad promedio sea mayor a cero).

¿Cómo se hace el proceso de Verificación?

Se parte de una especificación {**P**} **S** {**Q**}.

El proceso de verificación se basa en la definición de la precondición más débil: "**pmd"**, la **menos restrictiva,** que hace válida el reemplazo de P y define una nueva terna:

{**pmd**} **S** {Q}

Esto se lee "Precondición más débil de **S** respecto de **Q".**

Con la pmd podemos decir que el algoritmo (programa) es correcto si la precondición {P} es más fuerte que la pmd (P es subconjunto de la pmd), con lo cual se concluye que el algoritmo es correcto para las especificaciones P y Q.

Como regla general:

1. Se debe encontrar la precondición más débil {pmd} tal que {**pmd**} **S** {**Q**} es válido
2. Cualquier precondición {P} que implique {pmd} hace que el programa S sea correcto.

Esto es:

Si {P} $\Rightarrow$ {**pmd**} entonces el programa S es correcto para la terna {**P**} **S** {**Q**}.

Paso 1: ¿Cómo se calcula la pmd?

Se parte de las acciones de asignación y de la postcondición. Son acciones de la forma V = E, en donde V es una variable y E es una expresión o valor.

{ pmd } S { Q }

{ pmd } V=E { Q }

{ pmd } x=2*y+1 { x ≤15 } x, y $\in$ Z

Si S es una acción de la forma V=E con la postcondición {Q}, entonces la precondición más débil de S puede hallarse sustituyendo en Q todos los casos de V por E. Esto se simboliza $(Q)_{V}^{E}$

Esquemáticamente: {pmd} $\equiv$ $(Q)_{v}^{E}$

Ejemplo:

{pmd} x=2*y+1 {x ≤15} , x, y $\in$ Z

pmd $(Q)_{v}^{E} \Rightarrow (x \leq 15)_{x}^{2*y+1}$

$(2*y+1 \leq 15)$ *se reemplazó a x*

$(y \leq 7)$ *se hizo el pasaje de termino de 2 y de 1*

{pmd: y ≤ 7}

Paso 2: Analizar la implicación

Si {P} es {y == 3} debemos verificar que es subconjunto de la precondición más débil

{P} $\Rightarrow$ {pmd}

{P: y == 3} $\Rightarrow$ {pmd: y ≤ 7} esto es verdadero ya que P es subconjunto de pmd. La pmd es el conjunto de todos los números menores – iguales a 7 y 3 forma parte de ese conjunto. P es más fuerte.

¿Qué hacer cuando se tiene más de una acción?
Cuando se tiene más de una acción se deben buscar las pmd intermedias, comenzando desde la última acción hasta la primera.
Esto significa que para demostrar que la terna {P} A1; A2 {Q} es correcta, se debe encontrar el predicado intermedio {pmd1} que es postcondición de A1 y precondición de A2.

Para la forma: {P} acción1 acción2 {Q} equivalente a: {P} A1; A2 {Q}

¿Cómo se resuelve?
Paso 1: se encuentra la pmd
Para esto se construye la terna {*pmd1*} A2 {Q}
Se calcula la pmd1, cuando se obtiene esta se construye la siguiente terna {pmd} A1 {*pmd1*}
Se calcula la pmd y se va al paso siguiente

Paso 2: Analizar la implicación {P} ⇒ {pmd}
Determinar si P es subconjunto de pmd, dicho de otro modo, si P es más fuerte que pmd.
Con esto se verifica la correctitud el programa.

Recordar:

Dada una especificación {P} S {Q}, para verificar el código se requiere realizar dos pasos específicos:

1) Calcular la pmd: {pmd} ≡ (Q) $^{E}_{V}$

2) Analizar la implicación: P ⇒ pmd

Si se cumple la implicación, significa que la acción S esta verificada, esto implica que la terna de Hoare {P} S {Q} es correcta.