VERIFICACION DE PROGRAMAS

Programas secuenciales

UNSL

INTRODUCCIÓN

- Un **error** en el programa puede ser **muy costoso**:
  - a nivel económico
  - a nivel de vidas humanas

Es importante disponer de **métodos** que permitan comprobar que un programa cumple las especificaciones con que fue diseñado.

INTRODUCCIÓN

- **VERIFICACIÓN DE PROGRAMAS**: métodos que permiten comprobar de manera automática o semiautomática que cierto programa cumple con las especificaciones, es decir, hace lo que se espera de él, sin cometer nunca errores.

INTRODUCCIÓN

Para poder comprobar que un programa es correcto vamos a:

1. Definir un lenguaje de programación
2. Establecer con precisión su semántica

LOGICA DE HOARE

3. Utilizar un método deductivo para la lógica de Hoare y aplicarlo a la verificación de programas.

SINTAXIS

⌘ MICRO LENGUAJE DE PROGRMACIÓN

- Definimos un pequeño lenguaje que sólo tiene 3 instrucciones y 2 tipos de datos (enteros, booleanos).

expresión matemática	$E ::= n \mid x \mid (-E) \mid (E + E) \mid (E - E) \mid (E * E)$
expresión booleana	$B ::= \text{true} \mid \text{false} \mid !B \mid (B \,\&\, B) \mid (B \mid B) \mid (E == E) \mid (E \mathrel{!}= E)$
instrucción	$S ::= x := E \mid \text{if } B \text{ then} < (S) \left[\text{else } (S)\right] \mid \text{while } B \text{ to } (S) \mid S; S$

SINTAXIS

⌘ MICRO LENGUAJE DE PROGRMACIÓN

Ejemplo -fact1-

```
fact := 1;
i:= 0;
While(i != x) do (
i:= i +1 ;
fact := fact * i;
)
```

SINTAXIS

ESPECIFICACIÓN DE ESTADOS

- Dado el conjunto de variables que aparecen en un programa, un **estado** viene dado por la asignación de un valor a cada una de las variables.

Ejemplo: Si las variables son `x`, `i` y `fact`

$$\{x = 10 \wedge i = 4 \wedge fact = 24\}$$

$$\{i = 4\}$$ conjunto de estados

NOTA: cualquier expresión lógica representa un estado o un conjunto de estados.

{T} conjunto de todos los posibles estados

{⊥} no representa ningún estado (conjunto vacio)

SINTAXIS

TERNAS DE HOARE

- Dado un lenguaje que expresa los estados de un sistema y un lenguaje de programación , definimos un nuevo lenguaje formado por ternas de la forma:

$$\{\text{precondición}\}(\text{instrucción})\{\text{postcondición}\}$$

Terna de Hoare

EJEMPLO: $\{i = 4\}(\mathrm{i} := \mathrm{i} + 1)\{\mathrm{i} = 5\}$

$$\{i > 0\}(\mathrm{i} := \mathrm{i} + 1)\{\mathrm{i} > 1\}$$

NOTA: cuando tenemos una terna de Hoare en que el programa no esta definido todavía, se dice que tenemos una **especificación del programa**, dado que sólo se indica cuales son los requisitos que debe cumplir el programa.

$$\{x \geq 0\}(\mathrm{S})\{fact = x!\}$$

SINTAXIS

⌘ Variables de programa vs lógicas

Ejemplo: Queremos expresar que cualquiera que sea el valor inicial de i, la instrucción i:= i +1 hace q el valor de esta variable aumente en una unidad.

$$\{\mathrm{T}\}(\mathrm{i} := \mathrm{i}+1)\{i = i+1\}$$ NO

$$\forall a, \{i = a\}(\mathrm{i} := \mathrm{i}+1)\{i = a+1\}$$

La variable ***a*** que no aparece en la instrucción (no forma parte del programa) se denomina **variable lógica** y sirve para relacionar la precondición con la postcondición. Las **variables de programa** son las que aparecen en la instrucción S.

$$\{i = a\}(\mathrm{i} := \mathrm{i}+1)\{i = a+1\}$$ Se omite el cuantificador universal

SEMANTICA

⌘ CORRECCIÓN TOTAL

- Una terna de Hoare, {p}(S){q} puede considerarse una proposición y por tanto es posible asignarle un valor de verdad. Decimos que una terna de Hoare es cierta si todo sistema que parte de un estado (cualquiera) que satisface p pasa a un estado que satisface a q.

$$\models_{\text{tot}} \{p\}(\text{S})\{q\}$$

Esta propiedad, denominada **corrección total**, es muy importante en la práctica, pues la *verificación* del programa S consiste precisamente en demostrar que siempre que el sistema satisface ciertas condiciones (p) la ejecución de S hace que se satisfaga la condición que nos interesa (q).

$$\models_{\text{tot}} \{i = 4\}(\text{i} := \text{i} + 1)\{i = 5\}$$

SEMANTICA

⌘ CORRECCIÓN PARCIAL

- La **corrección parcial** significa que si un sistema que se encuentre en un estado que satisface la condición p, ejecuta el programa S, **si el programa termina**, llega al estado q.

$$\vDash_{\text{par}} \{p\}(\text{S})\{q\}$$

Nota: Un programa que no termina nunca siempre es parcialmente correcto (para toda p y toda q), pero nunca es totalmente correcto.

$$\vDash_{\text{par}} \{p\}(\text{while true do (i := 0)})\{q\}$$

$$\nvDash_{\text{tot}} \{p\}(\text{while true do (i := 0)})\{q\}$$

$$\vDash_{\text{par}} \{\bot\}(\text{S})\{q\} \qquad \vDash_{\text{par}} \{p\}(\text{S})\{\text{T}\}$$

SEMANTICA

⌘ CORRECCIÓN TOTAL

- En la práctica, demostrar la corrección total directamente suele ser bastante complicado. En estos casos, podemos resolver el problema dividiéndolo en dos subtareas:

  1. Demostrar que el programa, si termina, llega a q (corrección parcial), y
  2. Demostrar que el programa termina.

Nota: Las única instrucciones que pueden hacer que el programa no termine son los bucles `while`. Por eso, para demostrar que un programa termina basta demostrar que todos los bucles terminan.

SISTEMA DEDUCTIVO DE HOARE

El sistema deductivo de Hoare se basa en cinco axiomas, que sirven como reglas de deducción:

- Asignación

$$\vdash \{P(E)\}(\mathrm{x} := E)\{P(\mathrm{x})\}$$

- Condicional

$$\frac{\vdash \{p \wedge B\}(\mathrm{S}_1)\{q\} \qquad \vdash \{p \wedge \neg B\}(\mathrm{S}_2)\{q\}}{\vdash \{p\}(\mathit{if}\ B\ \mathtt{then}\ (\mathrm{S}_1)\ \mathtt{else}\ (\mathrm{S}_2))\{q\}}$$

- Bucle

$$\frac{\vdash \{p \wedge B\}(\mathrm{S})\{p\}}{\vdash \{p\}(\mathit{while}\ B\ \mathtt{do}\ (\mathrm{S}))\{p \wedge \neg B\}}$$

- Composición

$$\frac{\vdash \{p\}(\mathrm{S}_1)\{q\} \qquad \vdash \{q\}(\mathrm{S}_2)\{r\}}{\vdash \{p\}(\mathrm{S}_1;\mathrm{S}_2)\{r\}}$$

- Encadenamiento

$$\frac{\vdash \{p \to p'\} \quad \vdash \{p'\}(\mathrm{S})\{q'\} \quad \vdash \{q' \to q\}}{\vdash \{p\}(\mathrm{S})\{q\}}$$

SISTEMA DEDUCTIVO DE HOARE

Nota: A las reglas anteriores habría que añadir los axiomas propios del dominio. En el caso de nuestro microlenguaje el dominio es la aritmética entera, pues el único tipo de dato que admite son los enteros (aparte de expresiones booleanas, que forman parte de la lógica).

EJEMPLO:

$$(\forall x, x = x)$$

$$(\forall x(\forall y(\forall z, x = y \rightarrow x + z = y + z)))$$

En la verificación de programas se combinan dos sistemas deductivos, uno para razonar sobre las ternas de Hoare y otro para razonar sobre el dominio (la aritmética).

SISTEMA DEDUCTIVO DE HOARE

⌘ Regla de asignación

- Nos dice que si una condición (predicado) se cumple para la expresión *E* entonces se cumple también para x después de haber asignado a la variable x el valor *E* (x:=*E*)

$$\vdash \{P(E)\}(\mathrm{x} := E)\{P(\mathrm{x})\}$$

SISTEMA DEDUCTIVO DE HOARE

⌘ Regla de asignación (EJEMPLO)

$$\vdash \{a = 0\}(\mathrm{x} := a)\{\mathrm{x} = 0\}$$

Predicado *P* es “igual a cero” y la expresión *E* (el valor asignado a la variable) es “a”, por tanto *P(E)* es $a = 0$.

SISTEMA DEDUCTIVO DE HOARE

⌘ Regla de asignación (EJEMPLO)

$$\vdash \{?\}(\mathrm{x} := a+1)\{\mathrm{x}=0\}$$

$$\{a+1=0\}(\mathrm{x} := a+1)\{\mathrm{x}=0\}$$

$$\vdash \{x+y=7\}(\mathrm{z} := \mathrm{x}+\mathrm{y})\{\mathrm{z}=7\}$$

$$\vdash \{x+1=4\}(\mathrm{x} := \mathrm{x}+1)\{x=4\}$$

SISTEMA DEDUCTIVO DE HOARE

⌘ Regla del condicional

- Nos dice que si tenemos que demostrar una terna de la forma:

$$\{p\}(if\ B\ \texttt{then}\ (\mathrm{S}_1)\ \texttt{else}\ (\mathrm{S}_2))\{q\}$$

- PASO 1: Demostramos que, cuando partimos de la condición p y B es cierto, la ejecución S_1 garantiza la condición q.

$$\{p \wedge B\}(\mathrm{S}_1)\{q\}$$

- PASO 2: Demostramos que, cuando partimos de la condición p y B es falso, la ejecución S_2 también garantiza la condición q.

$$\{p \wedge \neg B\}(\mathrm{S}_2)\{q\}$$

SISTEMA DEDUCTIVO DE HOARE

⌘ Regla del condicional (EJEMPLO)

$$\frac{\vdash \{p \wedge B\}(\mathrm{S}_1)\{q\} \qquad \vdash \{p \wedge \neg B\}(\mathrm{S}_2)\{q\}}{\vdash \{p\}(if\ \ B\ \texttt{then}\ (\mathrm{S}_1)\ \texttt{else}\ (\mathrm{S}_2))\{q\}}$$

Para demostrar que:

$$\{x \neq 0\}(\text{if } (\mathrm{x} > 0) \text{ then } (\mathrm{y} := \mathrm{x}) \text{ else } (\mathrm{y} := \text{-x}))\{y > 0\}$$

basta demostrar que

$$\{x \neq 0 \wedge x > 0\}(\mathrm{y} := \mathrm{x})\{y > 0\}$$

y que

$$\{x \neq 0 \wedge \neg(x > 0)\}(\mathrm{y} := \text{-x})\{y > 0\}$$

SISTEMA DEDUCTIVO DE HOARE

⌘ Regla del condicional modificada

$$\{p\}(if\ B\ \texttt{then}\ (S_1)\ \texttt{else}\ (S_2))\{q\}$$

- Supongamos hemos encontrado dos condiciones *p1* y *p2* tales que

$$\{p_1\}(S_1)\{q\} \qquad \{p_2\}(S_2)\{q\}$$

- Definimos *p* asi: $p = (B \rightarrow p_1) \wedge (\neg B \rightarrow p_2)$

- Como $p \wedge B \rightarrow p_1$, por regla el encadenamiento tenemos

$$\{p \wedge B\}(S_1)\{q\}$$

- Análogamente $p \wedge \neg B \rightarrow p_2$

$$\{p \wedge \neg B\}(S_2)\{q\}$$

SISTEMA DEDUCTIVO DE HOARE

⌘ Regla del condicional modificada

$$\frac{\vdash \{p \wedge B\}(\mathrm{S}_1)\{q\} \qquad \vdash \{p \wedge \neg B\}(\mathrm{S}_2)\{q\}}{\vdash \{p\}(if\ \ B\ \texttt{then}\ (\mathrm{S}_1)\ \texttt{else}\ (\mathrm{S}_2))\{q\}}$$

- Como $p \wedge B \rightarrow p_1$, por regla el encadenamiento tenemos

$$\{p \wedge B\}(\mathrm{S}_1)\{q\}$$

- Análogamente $p \wedge \neg B \rightarrow p_2$

$$\{p \wedge \neg B\}(\mathrm{S}_2)\{q\}$$

$$\frac{\vdash \{p_1\}(\mathrm{S}_1)\{q\} \qquad \vdash \{p_2\}(\mathrm{S}_2)\{q\}}{\vdash \{(B \rightarrow p_1) \wedge (\neg B \rightarrow p_2)\}(if\ \ B\ \texttt{then}\ (\mathrm{S}_1)\ \texttt{else}\ (\mathrm{S}_2))\{q\}}$$

SISTEMA DEDUCTIVO DE HOARE

Regla del bucle

- **Def**: la condición p es una invariante para la instrucción "`while` ***B*** `do (`***S***`)`" si y sólo si se cumple

$$\vdash \{p \wedge B\}(\mathrm{S})\{p\}$$

- Esto nos permite concluir que

$$\frac{\vdash \{p \wedge B\}(\mathrm{S})\{p\}}{\vdash \{p\}(\texttt{while } B \texttt{ do } (\mathrm{S}))\{p\}}$$

- Por otro lado la instrucción `while` sólo termina cuando **B** es falsa.

$$\frac{\vdash \{p \wedge B\}(\mathrm{S})\{p\}}{\vdash \{p\}(\texttt{while } B \texttt{ do } (\mathrm{S}))\{p \wedge \neg B\}}$$

SISTEMA DEDUCTIVO DE HOARE

⌘ Regla del bucle (EJEMPLO)

Queremos demostrar que:

$$\vdash \{x \geq 0\}(\text{while } (\text{x!= 0}) \text{ do } (\text{x := x -1}))\{x = 0\}$$

Para ello, demostramos la expresión

$$\vdash \{x \geq 0 \wedge x \neq 0\}(\text{x := x -1})\{x \geq 0\}$$

que indica que "x ≥0" es una invariante para este bucle.
Aplicando la regla del bucle, con las equivalencias p = "x ≥ 0" , B ="x ≠ 0" y S ="x:=x-1" se obtiene que

$$\vdash \{x \geq 0\}(\text{while } (\text{x!= 0}) \text{ do } (\text{x := x -1}))\{x \geq 0 \wedge \neg(x \neq 0)\}$$

de donde se deduce la expresión que queríamos demostrar.

SISTEMA DEDUCTIVO DE HOARE

⌘ Regla de composición

- Si queremos demostrar $\{p\}(\mathrm{S}_1;\mathrm{S}_2)\{r\}$

- Tenemos que buscar una propiedad q tal que

$$\{p\}(\mathrm{S}_1)\{q\}$$

$$\{q\}(\mathrm{S}_2)\{r\}$$

SISTEMA DEDUCTIVO DE HOARE

⌘ Regla de composición (EJEMPLO)

$$\{x \geq 0\}(\mathrm{y}: = \mathrm{x} + 1;\mathrm{z}: = 2 * \mathrm{y})\{z \geq 0\}$$

- basta demostrar estas dos fórmulas

$$\{x \geq 0\}(\mathrm{y}: = \mathrm{x} + 1)\{y \geq 0\}$$

$$\{y \geq 0\}(\mathrm{z}: = 2 * \mathrm{y})\{z \geq 0\}$$

- o bien estas dos

$$\{x \geq 0\}(\mathrm{y}: = \mathrm{x} + 1)\{y \geq 1\}$$

$$\{y \geq 1\}(\mathrm{z}: = 2 * \mathrm{y})\{z \geq 0\}$$

SISTEMA DEDUCTIVO DE HOARE

⌘ Regla de encadenamiento

$$\frac{\vdash\{p \to p'\} \quad \vdash\{p'\}(\mathrm{S})\{q'\} \quad \vdash\{q' \to q\}}{\vdash\{p\}(\mathrm{S})\{q\}}$$

propiedad
$(p \to p)$

Se deducen las reglas:

$$\frac{\vdash\{p \to p'\} \qquad \vdash\{p'\}(\mathrm{S})\{q\}}{\vdash\{p\}(\mathrm{S})\{q\}}$$

$$\frac{\vdash\{p\}(\mathrm{S})\{q'\} \qquad \vdash\{q' \to q\}}{\vdash\{p\}(\mathrm{S})\{q\}}$$

VERIFICACIÓN PARCIAL DE PROGRAMAS

⌘ Composición y encadenamiento

Supongamos un programa S con instrucciones $S = S_1; \ldots; S_n$

$\{p\}$

S_1

$\{p_1\}$ [Justificación -1]

$\vdots$

$\{p_{n-1}\}$ [Justificación (n -1)]

S_n

q [Justificación – n]

Regla de composición

Definimos $p_0 = p$ y $p_n = q$. Para cada eslabón $\vdash \{p_{i-1}\}(S_i)\{p_i\}$

VERIFICACIÓN PARCIAL DE PROGRAMAS

⌘ Composición y encadenamiento

Puede pasar que la postcondición de S_i no coincida con la precondición de S_{i+1}

$\{p\}$

S_i

$\{p_{i,1}\}$ $[\text{Justificación - i,1}]$

$\{p_{i,2}\}$ $[\text{Justificación - i,2}]$

$\vdots$

$\{p_{i,k}\}$ $[\text{Justificación - i, k}]$

S_{i+1}

Regla de encadenamiento

Para cada par de proposiciones $\vdash p_{i,j} \rightarrow p_{i,j+1}$ se demuestra por algún axioma del dominio o por alguna propiedad de la lógica de predicados.

VERIFICACIÓN PARCIAL DE PROGRAMAS

⌘ Composición y encadenamiento

EJEMPLO: $\vdash \{T\}(x := 2; y;= 3*x+1)\{y = 7\}$

$\{T\}$

$\{2 = 2\}$ $[\forall x, x = x]$ **Sistema deductivo de la aritmética**

$x := 2$

$\{x = 2\}$ [Regla de asignación] **Sistema deductivo de Hoare**

$\{3x+1 = 7\}$ $[3*2+1 = 7]$ **Sistema deductivo de la aritmética**

$y := 3*x+1$

$\{y = 7\}$ [Regla de asignación] **Sistema deductivo de Hoare**

VERIFICACIÓN PARCIAL DE PROGRAMAS

⌘ Tratamiento de las asignaciones

La regla de la asignación puede expresarse así:

$\{P(E)\}$

$x := E$

$\{P(x)\}$ [Regla de asignación]

EJEMPLO: $\vdash \{T\}(x := 2; y; = 3 * x + 1)\{y = 7\}$

$\{T\}$

$x := 2$

$\{?\}$ [?]

$y := 3 * x + 1$

$\{y = 7\}$ [?]

VERIFICACIÓN PARCIAL DE PROGRAMAS

⌘ Tratamiento de las asignaciones

EJEMPLO: $\vdash \{T\}(x := 2; y;= 3*x+1)\{y = 7\}$

$\{T\}$

$x := 2$

$\{3*x+1 = 7\}$ $[?]$

$y := 3*x+1$

$\{y = 7\}$ $[\text{Regla de asignación}]$

VERIFICACIÓN PARCIAL DE PROGRAMAS

⌘ Tratamiento de las asignaciones

EJEMPLO: $\vdash \{T\}(x := 2; y; = 3 * x + 1)\{y = 7\}$

$\{T\}$

$\{3 * 2 + 1 = 7\}$ [?]

$x := 2$

$\{3 * x + 1 = 7\}$ [Regla de asignación]

$y := 3 * x + 1$

$\{y = 7\}$ [Regla de asignación]

VERIFICACIÓN PARCIAL DE PROGRAMAS

⌘ Tratamiento de las asignaciones

EJEMPLO: $\vdash \{T\}(x := 2; y; = 3*x+1)\{y = 7\}$

$\{T\}$

$\{3*2+1=7\}$ [Aritmética de numeros enteros]

$\vdash T \rightarrow 3*2+1=7$

$x := 2$

$\{3*x+1=7\}$ [Regla de asignación]

$y := 3*x+1$

$\{y=7\}$ [Regla de asignación]

VERIFICACIÓN PARCIAL DE PROGRAMAS

⌘ Tratamiento de las instr. condicionales

$\{(B \to p_1) \wedge (\neg B \to p_2)\}$

if B then (

$\{p_1\}$

S_1

$\{q\}$ [Justificación -1]

) else (

$\{p_2\}$

S_2

$\{q\}$ [Justificación - 2]

)

$\{q\}$

Aplicamos la regla del condicional modificada

VERIFICACIÓN PARCIAL DE PROGRAMAS

⌘ Tratamiento de las instr. condicionales

EJEMPLO: $\{x \neq 0\}$(if (x > 0) then (y := x) else (y := -x))$\{y > 0\}$

$\{x \neq 0\}$

$\{?\}$ $[?]$

if (x > 0) then (

$\{p_1\ ?\}$

y := x

$\{y > 0\}$ $[?]$

) else (

$\{p_2\ ?\}$

y := -x

$\{y > 0\})$ $[?]$

$\{y > 0\}$ $[?]$

VERIFICACIÓN PARCIAL DE PROGRAMAS

⌘ Tratamiento de las instr. condicionales

EJEMPLO: $\{x \neq 0\}(\text{if } (x > 0) \text{ then } (y := x) \text{ else } (y := -x))\{y > 0\}$

$\{x \neq 0\}$

$\{?\}$ $[?]$

if $(x > 0)$ then (

$\{x > 0\}$

$y := x$

$\{y > 0\}$ [Regla de asignación]

) else (

$\{(-x) > 0\}$

$y := -x$

$\{y > 0\})$ [Regla de asignación]

$\{y > 0\}$ $[?]$

VERIFICACIÓN PARCIAL DE PROGRAMAS

⌘ Tratamiento de las instr. condicionales

EJEMPLO: $\{x \neq 0\}$(if (x > 0) then (y := x) else (y := -x))$\{y > 0\}$

$\{x \neq 0\}$

$\{(x > 0 \rightarrow x > 0) \wedge (\neg(x > 0) \rightarrow (-x) > 0\}$ [?]

$(B \rightarrow p_1) \wedge (\neg B \rightarrow p_2)$

if (x > 0) then (

$\{x > 0\}$

y := x

$\{y > 0\}$ [Regla de asignación]

) else (

$\{(-x) > 0\}$

y := -x

$\{y > 0\})$ [Regla de asignación]

$\{y > 0\}$ [?]

VERIFICACIÓN PARCIAL DE PROGRAMAS

⌘ Tratamiento de las instr. condicionales

EJEMPLO: $\{x \neq 0\}$(if (x > 0) then (y := x) else (y := -x))$\{y > 0\}$

$\{x \neq 0\}$

$x \neq 0 \rightarrow (x > 0 \rightarrow x > 0) \wedge (\neg(x > 0) \rightarrow (-x) > 0)$

$\{(x > 0 \rightarrow x > 0) \wedge (\neg(x > 0) \rightarrow (-x) > 0\}$ [?]

if (x > 0) then (

$\{x > 0\}$

y := x

$\{y > 0\}$ [Regla de asignación]

) else (

$\{(-x) > 0\}$

y := -x

$\{y > 0\})$ [Regla de asignación]

$\{y > 0\}$ [Regla de asignación]

VERIFICACIÓN PARCIAL DE PROGRAMAS

⌘ Tratamiento de los bucles

Cuando nos encontramos con:

$\{?\}$

while B do (S)

$\{q\}$ $[?]$

Tenemos que encontrar una invariante que nos permite escribir:

$\{p\}$

while B do (

$\{p \wedge B\}$

S

$\{p\}$ [Justificación (de que p es una invariante)]

)

$\{p \wedge \neg B\}$ [Regla del bucle]

$\{q\}$ $[p \wedge \neg B \rightarrow q]$

VERIFICACIÓN PARCIAL DE PROGRAMAS

⌘ Tratamiento de los bucles

Para ello la invariante *p* debe satisfacer:

1. $\vdash \{p \wedge B\}(S)\{p\}$ (es la definición de invariante)

2. $\vdash p \wedge \neg B \rightarrow q$ para poder obtener la postcondición del bucle

3. Hace falta que p pueda deducirse a partir de la postcondición de la instrucción anterior al bucle.

VERIFICACIÓN PARCIAL DE PROGRAMAS

⌘ Tratamiento de los bucles

EJEMPLO (`fact1`)

```
fact := 1;
i := 0;
while (i != x) do (
        i := i + 1;
        fact := fact * i;
)
```

Buscamos una invariante p que cumpla:

$$\vdash \{p \wedge i \neq x\}(\mathrm{i} := \mathrm{i}+1; \mathrm{fact} := \mathrm{fact} * \mathrm{i};)\{\mathrm{p}\}$$

$$\vdash \; p \wedge \neg(i \neq x) \rightarrow \; fact = x!$$

Especificación

$$\{x \geq 0\}(\mathrm{S})\{fact = \mathrm{x}!\}$$

VERIFICACIÓN PARCIAL DE PROGRAMAS

⌘ Tratamiento de los bucles

EJEMPLO (`fact1`)

En todas las columnas se cumple:

$$fact = i!$$

Es una invariante?

iteración	x	i	$fact$	B
1^a	6	0	1	$\top$
2^a	6	1	1	$\top$
3^a	6	2	2	$\top$
4^a	6	3	6	$\top$
5^a	6	4	24	$\top$
6^a	6	5	120	$\top$
7^a	6	6	720	$\bot$

$\{fact = i! \wedge i \neq x\}$

i := i + 1;

$\{?\}$ $[?]$

fact := fact*i;

$\{fact = i!\}$ $[?]$

VERIFICACIÓN PARCIAL DE PROGRAMAS

⌘ Tratamiento de los bucles

EJEMPLO (`fact1`)

Aplicando 2 veces la regla de asignación nos queda:

$\{fact = i! \wedge i \neq x\}$

$\{fact = i!\}$ $[p \wedge q \rightarrow p]$

$\{fact * (i+1)! = (i+1)!\}$ $[\forall a, (a+1)! = (a+1) * a!]$

i := i + 1;

$\{fact * i = i!\}$ [Regla de asignación]

fact := fact * i;

$\{fact = i!\}$ [Regla de asignación]

$fact = i!$

Es una invariante

VERIFICACIÓN PARCIAL DE PROGRAMAS

⌘ Tratamiento de los bucles

EJEMPLO (`fact1`)

$\{x \geq 0\}$

$\{1 = 0!\}$ $[\text{Axioma (def. factorial)}]$

fact := 1; $[\text{Regla de asignación}]$

$\{fact = 0!\}$

i := 0;

$\{fact = i!\}$ $[\text{Regla de asignación}]$

while (i!= x) do (

$\{fact = i! \wedge i \neq x\}$

$\{fact = i!\}$ $[p \wedge q \rightarrow p]$

$\{fact * (i+1)! = (i+1)!\}$ $[\forall a, (a+1)! = (a+1) * a!]$

i := i +1;

$\{fact * i = i!\}$ $[\text{Regla de asignación}]$

fact := fact* i;

$\{fact = i!\}$ $[\text{Regla de asignación}]$

)

$\{fact = i! \wedge \neg(i \neq x)\}$ $[\text{Regla del bucle}]$

$\{fact = i! \wedge i = x\}$ $[\forall a, \forall b, a \neq b \leftrightarrow \neg(a = b)]$

$\{fact = x!\}$ $[\text{Regla de sustitución (prop. de la lógica)}]$

VERIFICACIÓN TOTAL DE PROGRAMAS

Para la corrección total es necesario sustituir la regla de bucle anterior:

⌘ Regla de bucle para corrección total

$$\vdash \frac{\{p \wedge B \wedge E \geq 0 \wedge E = a\}(\mathrm{S})\{p \wedge E \geq 0 \wedge E < a\}}{\vdash \{p \wedge E \geq 0\}(\texttt{while } B \texttt{ do } (\mathrm{S}))\{p \wedge \neg B\}}$$

La expresión E se denomina variante.

Atención:

- ✓ El **invariante** es una proposición p ⟶ $fact = i!$
- ✓ El **variante** es una expresión numérica E ⟶ $x - i$

Tanto p como E hacen referencia al valor de variables del programa.

VERIFICACIÓN TOTAL DE PROGRAMAS

⌘ Regla de bucle para corrección total

EJEMPLO (`fact1`)

```
fact := 1;
i := 0;
while (i != x) do (
        i := i + 1;
        fact := fact * i;
)
```

invariante → $fact = i!$

Aunque todavía no hemos encontrado E, sabemos que se cumple:

$$\vdash \{p \wedge B \wedge E \geq 0 \wedge E = a\}(S)\{\mathrm{p}\}$$

$$\vdash \{fact = i! \wedge i \neq x \wedge E \geq 0 \wedge E = a\}(\mathrm{i} := \mathrm{i}+1; \mathrm{fact} := \mathrm{fact} * \mathrm{i})\{fact = i!\}$$

VERIFICACIÓN TOTAL DE PROGRAMAS

⌘ Regla de bucle para corrección total

EJEMPLO (`fact1`)

```
fact := 1;
i := 0;
while (i != x) do (
        i := i + 1;
        fact := fact * i;
    )
```

variante → $x - i$

Debemos buscar una expresión E que cumpla:

$$\vdash \{p \wedge B \wedge E \geq 0 \wedge E = a\}(S)\{E \geq 0 \wedge E < a\}$$

$$\vdash \{i \neq x \wedge x - i \geq 0 \wedge x - i = a\}(\mathrm{i} := \mathrm{i} + 1; \mathrm{fact} := \mathrm{fact} * \mathrm{i})\{x - i \geq 0 \wedge x - i < a\}$$

Gracias por su atención!!

DUDAS?