

Tecnología de la Programación

Semántica Operacional

David Cabrero Souto

Facultad de Informática
Universidade da Coruña

Curso 2007/2008

- Recordar descriptores BOE:
 - Diseño de algoritmos
 - Análisis de algoritmos
 - **Lenguajes de programación**
 - Diseño de programas: Descomposición modular y documentación
 - **Técnicas de verificación**
 - **Pruebas de programas**
- Pruebas de programas: *validación*
 - Ejecución de un conjunto de tests generados sintéticamente o manualmente.
- Prueba formal de propiedades: *verificación*
 - Demostración formal de propiedades .
 - Necesitamos una definición formal del significado (**semántica**) de los lenguajes de programación.

- Recordar descriptores BOE:
 - Diseño de algoritmos
 - Análisis de algoritmos
 - **Lenguajes de programación**
 - Diseño de programas: Descomposición modular y documentación
 - **Técnicas de verificación**
 - **Pruebas de programas**
- Pruebas de programas: *validación*
 - Ejecución de un conjunto de tests generados sintéticamente o manualmente.
- Prueba formal de propiedades: *verificación*
 - Demostración formal de propiedades .
 - Necesitamos una definición formal del significado (**semántica**) de los lenguajes de programación.

- Recordar descriptores BOE:
 - Diseño de algoritmos
 - Análisis de algoritmos
 - **Lenguajes de programación**
 - Diseño de programas: Descomposición modular y documentación
 - **Técnicas de verificación**
 - **Pruebas de programas**
- Pruebas de programas: *validación*
 - Ejecución de un conjunto de tests generados sintética o manualmente.
- Prueba formal de propiedades: *verificación*
 - Demostración formal de propiedades .
 - Necesitamos una definición formal del significado (**semántica**) de los lenguajes de programación.

- Recordar descriptores BOE:
 - Diseño de algoritmos
 - Análisis de algoritmos
 - **Lenguajes de programación**
 - Diseño de programas: Descomposición modular y documentación
 - **Técnicas de verificación**
 - **Pruebas de programas**
- Pruebas de programas: *validación*
 - Ejecución de un conjunto de tests generados sintética o manualmente.
- Prueba formal de propiedades: *verificación*
 - Demostración formal de propiedades .
 - Necesitamos una definición formal del significado (**semántica**) de los lenguajes de programación.

- Recordar descriptores BOE:
 - Diseño de algoritmos
 - Análisis de algoritmos
 - **Lenguajes de programación**
 - Diseño de programas: Descomposición modular y documentación
 - **Técnicas de verificación**
 - **Pruebas de programas**
- Pruebas de programas: *validación*
 - Ejecución de un conjunto de tests generados sintética o manualmente.
- Prueba formal de propiedades: *verificación*
 - Demostración formal (base matemática) de propiedades .
 - Necesitamos una definición formal del significado (**semántica**) de los lenguajes de programación.

- Recordar descriptores BOE:
 - Diseño de algoritmos
 - Análisis de algoritmos
 - **Lenguajes de programación**
 - Diseño de programas: Descomposición modular y documentación
 - **Técnicas de verificación**
 - **Pruebas de programas**
- Pruebas de programas: *validación*
 - Ejecución de un conjunto de tests generados sintética o manualmente.
- Prueba formal de propiedades: *verificación*
 - Demostración formal (base matemática) de propiedades (para todos los casos).
 - Necesitamos una definición formal del significado (**semántica**) de los lenguajes de programación.

- Recordar descriptores BOE:
 - Diseño de algoritmos
 - Análisis de algoritmos
 - **Lenguajes de programación**
 - Diseño de programas: Descomposición modular y documentación
 - **Técnicas de verificación**
 - **Pruebas de programas**
- Pruebas de programas: *validación*
 - Ejecución de un conjunto de tests generados sintética o manualmente.
- Prueba formal de propiedades: *verificación*
 - Demostración formal (base matemática) de propiedades (para todos los casos).
 - Necesitamos una definición formal del significado (**semántica**) de los lenguajes de programación.

- IMP (IMPerativo)
- Estructuras básicas:
 - Secuencia
 - Alternativa
 - Repetición
- Cambio de estado: variables y asignación.
- Tipos básicos: enteros, strings.
- Tipos compuestos: arrays.
- Funciones.

IMP. Sintaxis aproximada

```
IMP ::= 'eof'
      | CMDS 'eof'

CMDS ::= CMD ';'
      | CMD ';' CMDS

CMD ::= 'skip'
      | Type Variable ':=' EXP
      | Variable ':=' EXP
      | Type Variable '[' Exp ']' ':=' EXP
      | Variable '[' Exp ']' ':=' EXP
      | 'if' EXP 'then' CMDS 'else' CMDS 'fi'
      | 'while' EXP 'do' CMDS 'done'
      | 'print' EXP
```

Semántica operacional

- Define la semántica de un lenguaje de programación en función de los *cambios de estado* que producen cada una de las instrucciones del lenguaje.
- No es adecuado para todo tipo de lenguajes.
- El estado se representa mediante un modelo o abstracción.

Conjuntos sintácticos asociados con IMP.

- números N
- Booleanos $T = \{true, false\}$
- Posiciones de memoria Loc
- Expresiones aritméticas $Aexp$
- Expresiones booleanas $Bexp$
- Comandos Com

Conjuntos sintácticos asociados con IMP.

- números N
- Booleanos $T = \{true, false\}$
- Posiciones de memoria Loc
- Expresiones aritméticas $Aexp$
- Expresiones booleanas $Bexp$
- Comandos Com

Conjuntos sintácticos asociados con IMP.

- números N
- Booleanos $T = \{true, false\}$
- Posiciones de memoria Loc
- Expresiones aritméticas $Aexp$
- Expresiones booleanas $Bexp$
- Comandos Com

Conjuntos sintácticos asociados con IMP.

- números N
- Booleanos $T = \{true, false\}$
- Posiciones de memoria Loc
- Expresiones aritméticas $Aexp$
- Expresiones booleanas $Bexp$
- Comandos Com

Conjuntos sintácticos asociados con IMP.

- números N
- Booleanos $T = \{true, false\}$
- Posiciones de memoria Loc
- Expresiones aritméticas $Aexp$
- Expresiones booleanas $Bexp$
- Comandos Com

Conjuntos sintácticos asociados con IMP.

- números N
- Booleanos $T = \{true, false\}$
- Posiciones de memoria Loc
- Expresiones aritméticas $Aexp$
- Expresiones booleanas $Bexp$
- Comandos Com

Convenciones

- n, m son variables en N
- X, Y son variables en Loc
- a_i son variables en $Aexp$
- b_i son variables en $Bexp$
- c_i son variables en Com

$n, m \in N$
 $X, Y \in Loc$
 $a_i \in Aexp$
 $b_i \in Bexp$
 $c_i \in Com$

Convenciones

- n, m son variables en N
- X, Y son variables en Loc
- a_i son variables en $Aexp$
- b_i son variables en $Bexp$
- c_i son variables en Com

$n, m \in N$
 $X, Y \in Loc$
 $a_i \in Aexp$
 $b_i \in Bexp$
 $c_i \in Com$

Convenciones

- n, m son variables en N
- X, Y son variables en Loc
- a_i son variables en $Aexp$
- b_i son variables en $Bexp$
- c_i son variables en Com

$n, m \in N$
 $X, Y \in Loc$
 $a_i \in Aexp$
 $b_i \in Bexp$
 $c_i \in Com$

Convenciones

- n, m son variables en N
- X, Y son variables en Loc
- a_i son variables en $Aexp$
- b_i son variables en $Bexp$
- c_i son variables en Com

$n, m \in N$
 $X, Y \in Loc$
 $a_i \in Aexp$
 $b_i \in Bexp$
 $c_i \in Com$

Convenciones

- n, m son variables en N
- X, Y son variables en Loc
- a_i son variables en $Aexp$
- b_i son variables en $Bexp$
- c_i son variables en Com

$n, m \in N$
 $X, Y \in Loc$
 $a_i \in Aexp$
 $b_i \in Bexp$
 $c_i \in Com$

- Expresiones aritméticas

$$\begin{array}{l} Aexp ::= n \\ \quad | X \\ \quad | a_0 + a_1 \\ \quad | a_0 - a_1 \\ \quad | a_0 * a_1 \end{array}$$

- Expresiones booleanas

$Bexp ::=$

- true**
- false**
- $a_0 = a_1$
- $a_0 \leq a_1$
- not** b
- b_0 **and** b_1
- b_0 **or** b_1

- Instrucciones

Comm ::= **skip**
| $X := a$
| $c_0; c_1$
| **if** b **then** c_0 **else** c_1 **fi**
| **while** b **do** c **done**

- El conjunto de estados Σ está formado por las funciones:

$$\sigma : \mathbf{Loc} \rightarrow \mathbf{N}$$

(Sólo variables enteras)

- Dado un estado σ , representaremos el valor de la posición X como:

$$\sigma(X)$$

- Y la extensión de un estado $\sigma[X \leftarrow m]$:

$$\sigma[X \leftarrow m](Y) = \begin{cases} m & Y = X, \\ \sigma(Y) & \text{si } Y \neq X. \end{cases}$$

Indistintamente $\sigma[X \leftarrow m]$ ó $\sigma[m/X]$

- Evaluación de una expresión aritmética a en un estado σ
 $\langle a, \sigma \rangle \rightarrow n$
- Evaluación de una expresión booleana b en un estado σ
 $\langle b, \sigma \rangle \rightarrow \{\mathbf{true}, \mathbf{false}\}$
- Ejecución de una instrucción c en un estado σ
 $\langle c, \sigma \rangle \rightarrow \sigma'$

- Si las premisas son ciertas, se puede deducir la conclusión.

$$\frac{\langle \textit{premisas} \rangle}{\langle \textit{conclusion} \rangle}$$

- $\frac{F_1 \dots F_n}{G_1 \dots G_m} \quad F_1 \dots F_n \rightarrow G_1 \dots G_m$

- Las usaremos para definir la semántica operacional.

Evaluación de expresiones aritméticas

- Constantes

$$\frac{}{\langle n, \sigma \rangle \rightarrow n}$$

- Variables

$$\frac{}{\langle X, \sigma \rangle \rightarrow \sigma(X)}$$

- Suma

$$\frac{\langle a_0, \sigma \rangle \rightarrow n \quad \langle a_1, \sigma \rangle \rightarrow m}{\langle a_0 + a_1, \sigma \rangle \rightarrow n + m}$$

- Resta

$$\frac{\langle a_0, \sigma \rangle \rightarrow n \quad \langle a_1, \sigma \rangle \rightarrow m}{\langle a_0 - a_1, \sigma \rangle \rightarrow n - m}$$

- Multiplicación

$$\frac{\langle a_0, \sigma \rangle \rightarrow n \quad \langle a_1, \sigma \rangle \rightarrow m}{\langle a_0 * a_1, \sigma \rangle \rightarrow n * m}$$

Evaluación de expresiones aritméticas

- Constantes

$$\frac{}{\langle n, \sigma \rangle \rightarrow n}$$

- Variables

$$\frac{}{\langle X, \sigma \rangle \rightarrow \sigma(X)}$$

- Suma

$$\frac{\langle a_0, \sigma \rangle \rightarrow n \quad \langle a_1, \sigma \rangle \rightarrow m}{\langle a_0 + a_1, \sigma \rangle \rightarrow n + m}$$

- Resta

$$\frac{\langle a_0, \sigma \rangle \rightarrow n \quad \langle a_1, \sigma \rangle \rightarrow m}{\langle a_0 - a_1, \sigma \rangle \rightarrow n - m}$$

- Multiplicación

$$\frac{\langle a_0, \sigma \rangle \rightarrow n \quad \langle a_1, \sigma \rangle \rightarrow m}{\langle a_0 * a_1, \sigma \rangle \rightarrow n * m}$$

Evaluación de expresiones aritméticas

- Constantes

$$\frac{}{\langle n, \sigma \rangle \rightarrow n}$$

- Variables

$$\frac{}{\langle X, \sigma \rangle \rightarrow \sigma(X)}$$

- Suma

$$\frac{\langle a_0, \sigma \rangle \rightarrow n \quad \langle a_1, \sigma \rangle \rightarrow m}{\langle a_0 + a_1, \sigma \rangle \rightarrow n + m}$$

- Resta

$$\frac{\langle a_0, \sigma \rangle \rightarrow n \quad \langle a_1, \sigma \rangle \rightarrow m}{\langle a_0 - a_1, \sigma \rangle \rightarrow n - m}$$

- Multiplicación

$$\frac{\langle a_0, \sigma \rangle \rightarrow n \quad \langle a_1, \sigma \rangle \rightarrow m}{\langle a_0 * a_1, \sigma \rangle \rightarrow n * m}$$

Evaluación de expresiones aritméticas

- Constantes

$$\frac{}{\langle n, \sigma \rangle \rightarrow n}$$

- Variables

$$\frac{}{\langle X, \sigma \rangle \rightarrow \sigma(X)}$$

- Suma

$$\frac{\langle a_0, \sigma \rangle \rightarrow n \quad \langle a_1, \sigma \rangle \rightarrow m}{\langle a_0 + a_1, \sigma \rangle \rightarrow n + m}$$

- Resta

$$\frac{\langle a_0, \sigma \rangle \rightarrow n \quad \langle a_1, \sigma \rangle \rightarrow m}{\langle a_0 - a_1, \sigma \rangle \rightarrow n - m}$$

- Multiplicación

$$\frac{\langle a_0, \sigma \rangle \rightarrow n \quad \langle a_1, \sigma \rangle \rightarrow m}{\langle a_0 * a_1, \sigma \rangle \rightarrow n * m}$$

Evaluación de expresiones aritméticas

- Constantes

$$\overline{\langle n, \sigma \rangle \rightarrow n}$$

- Variables

$$\overline{\langle X, \sigma \rangle \rightarrow \sigma(X)}$$

- Suma

$$\frac{\langle a_0, \sigma \rangle \rightarrow n \quad \langle a_1, \sigma \rangle \rightarrow m}{\langle a_0 + a_1, \sigma \rangle \rightarrow n + m}$$

- Resta

$$\frac{\langle a_0, \sigma \rangle \rightarrow n \quad \langle a_1, \sigma \rangle \rightarrow m}{\langle a_0 - a_1, \sigma \rangle \rightarrow n - m}$$

- Multiplicación

$$\frac{\langle a_0, \sigma \rangle \rightarrow n \quad \langle a_1, \sigma \rangle \rightarrow m}{\langle a_0 * a_1, \sigma \rangle \rightarrow n * m}$$

Evaluación de expresiones booleanas (I)

- Constantes

$$\frac{}{\langle \mathbf{true}, \sigma \rangle \rightarrow \mathbf{true}}$$

$$\frac{}{\langle \mathbf{false}, \sigma \rangle \rightarrow \mathbf{false}}$$

- Comparación aritmética

$$\frac{\langle a_0, \sigma \rangle \rightarrow n \quad \langle a_1, \sigma \rangle \rightarrow m}{\langle a_0 = a_1, \sigma \rangle \rightarrow \mathbf{true}} \quad n = m$$

$$\frac{\langle a_0, \sigma \rangle \rightarrow n \quad \langle a_1, \sigma \rangle \rightarrow m}{\langle a_0 = a_1, \sigma \rangle \rightarrow \mathbf{false}} \quad n \neq m$$

$$\frac{\langle a_0, \sigma \rangle \rightarrow n \quad \langle a_1, \sigma \rangle \rightarrow m}{\langle a_0 \leq a_1, \sigma \rangle \rightarrow \mathbf{true}} \quad n \leq m$$

$$\frac{\langle a_0, \sigma \rangle \rightarrow n \quad \langle a_1, \sigma \rangle \rightarrow m}{\langle a_0 \leq a_1, \sigma \rangle \rightarrow \mathbf{false}} \quad n > m$$

Evaluación de expresiones booleanas (I)

- Constantes

$$\frac{}{\langle \mathbf{true}, \sigma \rangle \rightarrow \mathbf{true}}$$

$$\frac{}{\langle \mathbf{false}, \sigma \rangle \rightarrow \mathbf{false}}$$

- Comparación aritmética

$$\frac{\langle a_0, \sigma \rangle \rightarrow n \quad \langle a_1, \sigma \rangle \rightarrow m}{\langle a_0 = a_1, \sigma \rangle \rightarrow \mathbf{true}} \quad n = m$$

$$\frac{\langle a_0, \sigma \rangle \rightarrow n \quad \langle a_1, \sigma \rangle \rightarrow m}{\langle a_0 = a_1, \sigma \rangle \rightarrow \mathbf{false}} \quad n \neq m$$

$$\frac{\langle a_0, \sigma \rangle \rightarrow n \quad \langle a_1, \sigma \rangle \rightarrow m}{\langle a_0 \leq a_1, \sigma \rangle \rightarrow \mathbf{true}} \quad n \leq m$$

$$\frac{\langle a_0, \sigma \rangle \rightarrow n \quad \langle a_1, \sigma \rangle \rightarrow m}{\langle a_0 \leq a_1, \sigma \rangle \rightarrow \mathbf{false}} \quad n > m$$

- Negación

$$\frac{\langle b, \sigma \rangle \rightarrow \mathbf{true}}{\langle \mathbf{not } b, \sigma \rangle \rightarrow \mathbf{false}}$$

$$\frac{\langle b, \sigma \rangle \rightarrow \mathbf{false}}{\langle \mathbf{not } b, \sigma \rangle \rightarrow \mathbf{true}}$$

- Conectivas lógicas

$$\frac{\langle b_0, \sigma \rangle \rightarrow t_0 \quad \langle b_1, \sigma \rangle \rightarrow t_1}{\langle b_0 \mathbf{and } b_1, \sigma \rangle \rightarrow t} \quad t = t_0 \wedge t_1$$

$$\frac{\langle b_0, \sigma \rangle \rightarrow t_0 \quad \langle b_1, \sigma \rangle \rightarrow t_1}{\langle b_0 \mathbf{or } b_1, \sigma \rangle \rightarrow t} \quad t = t_0 \vee t_1$$

- Negación

$$\frac{\langle b, \sigma \rangle \rightarrow \mathbf{true}}{\langle \mathbf{not } b, \sigma \rangle \rightarrow \mathbf{false}}$$

$$\frac{\langle b, \sigma \rangle \rightarrow \mathbf{false}}{\langle \mathbf{not } b, \sigma \rangle \rightarrow \mathbf{true}}$$

- Conectivas lógicas

$$\frac{\langle b_0, \sigma \rangle \rightarrow t_0 \quad \langle b_1, \sigma \rangle \rightarrow t_1}{\langle b_0 \mathbf{and } b_1, \sigma \rangle \rightarrow t} \quad t = t_0 \wedge t_1$$

$$\frac{\langle b_0, \sigma \rangle \rightarrow t_0 \quad \langle b_1, \sigma \rangle \rightarrow t_1}{\langle b_0 \mathbf{or } b_1, \sigma \rangle \rightarrow t} \quad t = t_0 \vee t_1$$

- Implementación habitual

$$\frac{\langle b_0, \sigma \rangle \rightarrow \mathbf{false}}{\langle b_0 \mathbf{and} b_1, \sigma \rangle \rightarrow \mathbf{false}}$$
$$\frac{\langle b_0, \sigma \rangle \rightarrow \mathbf{true} \quad \langle b_1, \sigma \rangle \rightarrow \mathbf{false}}{\langle b_0 \mathbf{and} b_1, \sigma \rangle \rightarrow \mathbf{false}}$$
$$\frac{\langle b_0, \sigma \rangle \rightarrow \mathbf{true} \quad \langle b_1, \sigma \rangle \rightarrow \mathbf{true}}{\langle b_0 \mathbf{and} b_1, \sigma \rangle \rightarrow \mathbf{true}}$$

- Skip

$$\frac{}{\langle \mathbf{skip}, \sigma \rangle \rightarrow \sigma}$$

- Asignación

$$\frac{\langle a, \sigma \rangle \rightarrow m}{\langle X := a, \sigma \rangle \rightarrow \sigma[X \leftarrow m]}$$

- Secuencia

$$\frac{\langle c_0, \sigma \rangle \rightarrow \sigma'' \quad \langle c_1, \sigma'' \rangle \rightarrow \sigma'}{\langle c_0; c_1, \sigma \rangle \rightarrow \sigma'}$$

- Skip

$$\frac{}{\langle \mathbf{skip}, \sigma \rangle \rightarrow \sigma}$$

- Asignación

$$\frac{\langle a, \sigma \rangle \rightarrow m}{\langle X := a, \sigma \rangle \rightarrow \sigma[X \leftarrow m]}$$

- Secuencia

$$\frac{\langle c_0, \sigma \rangle \rightarrow \sigma'' \quad \langle c_1, \sigma'' \rangle \rightarrow \sigma'}{\langle c_0; c_1, \sigma \rangle \rightarrow \sigma'}$$

- Skip

$$\frac{}{\langle \mathbf{skip}, \sigma \rangle \rightarrow \sigma}$$

- Asignación

$$\frac{\langle a, \sigma \rangle \rightarrow m}{\langle X := a, \sigma \rangle \rightarrow \sigma[X \leftarrow m]}$$

- Secuencia

$$\frac{\langle c_0, \sigma \rangle \rightarrow \sigma'' \quad \langle c_1, \sigma'' \rangle \rightarrow \sigma'}{\langle c_0; c_1, \sigma \rangle \rightarrow \sigma'}$$

- If

$$\frac{\langle b, \sigma \rangle \rightarrow \text{true} \quad \langle c_0, \sigma \rangle \rightarrow \sigma'}{\langle \text{if } b \text{ then } c_0 \text{ else } c_1 \text{ fi}, \sigma \rangle \rightarrow \sigma'}$$

$$\frac{\langle b, \sigma \rangle \rightarrow \text{false} \quad \langle c_1, \sigma \rangle \rightarrow \sigma'}{\langle \text{if } b \text{ then } c_0 \text{ else } c_1 \text{ fi}, \sigma \rangle \rightarrow \sigma'}$$

- While

$$\frac{\langle b, \sigma \rangle \rightarrow \text{false}}{\langle \text{while } b \text{ do } c \text{ done}, \sigma \rangle \rightarrow \sigma}$$

$$\frac{\langle b, \sigma \rangle \rightarrow \text{true} \quad \langle c, \sigma \rangle \rightarrow \sigma'' \quad \langle \text{while } b \text{ do } c \text{ done}, \sigma'' \rangle \rightarrow \sigma'}{\langle \text{while } b \text{ do } c \text{ done}, \sigma \rangle \rightarrow \sigma'}$$

- If

$$\frac{\langle b, \sigma \rangle \rightarrow \text{true} \quad \langle c_0, \sigma \rangle \rightarrow \sigma'}{\langle \text{if } b \text{ then } c_0 \text{ else } c_1 \text{ fi}, \sigma \rangle \rightarrow \sigma'}$$

$$\frac{\langle b, \sigma \rangle \rightarrow \text{false} \quad \langle c_1, \sigma \rangle \rightarrow \sigma'}{\langle \text{if } b \text{ then } c_0 \text{ else } c_1 \text{ fi}, \sigma \rangle \rightarrow \sigma'}$$

- While

$$\frac{\langle b, \sigma \rangle \rightarrow \text{false}}{\langle \text{while } b \text{ do } c \text{ done}, \sigma \rangle \rightarrow \sigma}$$

$$\frac{\langle b, \sigma \rangle \rightarrow \text{true} \quad \langle c, \sigma \rangle \rightarrow \sigma'' \quad \langle \text{while } b \text{ do } c \text{ done}, \sigma'' \rangle \rightarrow \sigma'}{\langle \text{while } b \text{ do } c \text{ done}, \sigma \rangle \rightarrow \sigma'}$$

Derivaciones

- *Derivación*. Secuencia de aplicación de reglas.
- *Axioma*. Regla sin premisas.
- *Instancia de un regla*. Sustituimos las metavariables por valores concretos.
- Las derivaciones representan ejecuciones de las instrucciones o evaluaciones de las expresiones.
- Ejemplo. $a \equiv (X + 5) + (7 + 9)$ y $\sigma_0 = \{X \leftarrow 0\}$
 $\dot{?} \langle a, \sigma_0 \rangle ?$

$$\frac{\frac{\langle X, \sigma_0 \rangle \rightarrow 0}{\langle (X + 5), \sigma_0 \rangle \rightarrow 5} \quad \frac{\langle 5, \sigma_0 \rangle \rightarrow 5}{\langle (X + 5), \sigma_0 \rangle \rightarrow 5} \quad \frac{\langle 7, \sigma_0 \rangle \rightarrow 7}{\langle (7 + 9), \sigma_0 \rangle \rightarrow 16} \quad \frac{\langle 9, \sigma_0 \rangle \rightarrow 9}{\langle (7 + 9), \sigma_0 \rangle \rightarrow 16}}{\langle (X + 5) + (7 + 9), \sigma_0 \rangle \rightarrow 21}$$

- Si $w \equiv \text{while true do skip}$,
¿ $\forall \sigma, \exists \sigma' t.q. \langle w, \sigma \rangle \rightarrow \sigma'$?

- ¿ Son iguales ?

skip

~

```
b := 1 > 2;  
x := 1;  
while b do  
  x := x * 2;  
  b := x < 500;  
done
```

- Después de este código, ¿ X es par ?

```
while x < 3000 do  
  x := x * 2;  
done
```

Relaciones de equivalencia

- $a_0 \sim a_1$ **iff**
($\forall n \in \mathcal{N}, \forall \sigma \in \Sigma, \langle a_0, \sigma \rangle \rightarrow n \iff \langle a_1, \sigma \rangle \rightarrow n$)
- $b_0 \sim b_1$ **iff**
($\forall t \forall \sigma \in \Sigma, \langle b_0, \sigma \rangle \rightarrow t \iff \langle b_1, \sigma \rangle \rightarrow t$)
- $c_0 \sim c_1$ **iff**
($\forall \sigma, \sigma' \in \Sigma, \langle c_0, \sigma \rangle \rightarrow \sigma' \iff \langle c_1, \sigma \rangle \rightarrow \sigma'$)

- Proposición

Sea $w \equiv \mathbf{while\ } b \mathbf{ do\ } c \mathbf{ done}$ con $b \in \mathbf{Bexp}$, $c \in \mathbf{Comm}$
entonces:

$w \sim \mathbf{if\ } b \mathbf{ then\ } c; w \mathbf{ else\ skip\ fi}$

- Demostración.

- Método menos laborioso: *inducción de reglas*.

Inducción matemática

- Sea $P(n)$ una propiedad de los números naturales. El principio de inducción matemática dice que para demostrar que $P(n)$ es cierto para todos los números naturales es suficiente con demostrar que:
 - $P(0)$ es cierto.
 - Si $P(m)$ es cierto, entonces también lo es $P(m+1)$, para cualquier número natural m .
- Es decir,

$$(P(0) \ \& \ (\forall m \in \omega. P(m) \Rightarrow P(m+1))) \Rightarrow \forall n \in \omega. P(n)$$

- Ejercicio. Demostrar que la propiedad $P(n)$ se cumple para los números naturales.

$$P(n) \iff \sum_{i=1}^n (2i - 1) = n^2$$

- ¿Cómo demostramos que la evaluación de expresiones aritméticas en IMP es determinista ?

$$\langle a, \sigma \rangle \rightarrow m \quad \& \quad \langle a, \sigma \rangle \rightarrow m' \Rightarrow m = m'$$

- Sea $P(a)$ una propiedad de las expresiones aritméticas. Para demostrar que $(P(a))$ se cumple para todas las expresiones aritméticas es suficiente con demostrar que:
 - Se cumple para todos los número $P(m)$.
 - Se cumple para todas la variables $P(X)$.
 - Para todas la expresiones aritméticas a_0, a_1 , si $P(a_0)$ y $P(a_1)$ se cumplen, entonces también se cumple $P(a_0 + a_1)$.
 - Para todas la expresiones aritméticas a_0, a_1 , si $P(a_0)$ y $P(a_1)$ se cumplen, entonces también se cumple $P(a_0 - a_1)$.
 - Para todas la expresiones aritméticas a_0, a_1 , si $P(a_0)$ y $P(a_1)$ se cumplen, entonces también se cumple $P(a_0 \times a_1)$.

- ¿Cómo demostramos que la evaluación de expresiones aritméticas en IMP es determinista ?

$$\langle a, \sigma \rangle \rightarrow m \quad \& \quad \langle a, \sigma \rangle \rightarrow m' \Rightarrow m = m'$$

- Sea $P(a)$ una propiedad de las expresiones aritméticas. Para demostrar que $(P(a))$ se cumple para todas las expresiones aritméticas es suficiente con demostrar que:
 - Se cumple para todos los número $P(m)$.
 - Se cumple para todas la variables $P(X)$.
 - Para todas la expresiones aritméticas a_0, a_1 , si $P(a_0)$ y $P(a_1)$ se cumplen, entonces también se cumple $P(a_0 + a_1)$.
 - Para todas la expresiones aritméticas a_0, a_1 , si $P(a_0)$ y $P(a_1)$ se cumplen, entonces también se cumple $P(a_0 - a_1)$.
 - Para todas la expresiones aritméticas a_0, a_1 , si $P(a_0)$ y $P(a_1)$ se cumplen, entonces también se cumple $P(a_0 \times a_1)$.

$(\forall m \in N. P(m))$

& $(\forall X \in Loc. P(X))$

& $(\forall a_0, a_1 \in Aexp. P(a_0) \text{ y } P(a_1) \Rightarrow P(a_0 + a_1))$

& $(\forall a_0, a_1 \in Aexp. P(a_0) \text{ y } P(a_1) \Rightarrow P(a_0 - a_1))$

& $(\forall a_0, a_1 \in Aexp. P(a_0) \text{ y } P(a_1) \Rightarrow P(a_0 * a_1))$

$\Rightarrow \forall a \in Aexp. P(a)$

Proposición Para toda expresión aritmética a y números m y m'

$$\langle a, \sigma \rangle \rightarrow m \quad \& \quad \langle a, \sigma \rangle \rightarrow m' \Rightarrow m = m'$$

Demostración

Inducción bien fundada

- Las anteriores son casos particulares.

Relación bien fundada Es una relación binaria \prec sobre un conjunto A tal que no hay cadenas descendientes infinitas $\dots \prec a_j \prec \dots \prec a_1 \prec a_0$

Si $a \prec b$ decimos que a es un *predecesor* de b .

Es *irreflexiva*. $\nexists a/a \prec a$

$$a \preceq b \iff a = b \text{ o } a \prec b$$

Proposición Sea \prec una relación binaria en el conjunto A . La relación \prec está bien fundada iff todos los subconjuntos no vacíos Q de A tienen un elemento minimal, i.e. un elemento m tal que

$$m \in Q \& \forall b \prec m. b \notin Q$$

Inducción bien fundada

- Las anteriores son casos particulares.

Relación bien fundada Es una relación binaria \prec sobre un conjunto A tal que no hay cadenas descendientes infinitas $\cdots \prec a_j \prec \cdots \prec a_1 \prec a_0$

Si $a \prec b$ decimos que a es un *predecesor* de b .

Es *irreflexiva*. $\nexists a/a \prec a$

$$a \preceq b \iff a = b \text{ o } a \prec b$$

Proposición Sea \prec una relación binaria en el conjunto A . La relación \prec está bien fundada iff todos los subconjuntos no vacíos Q de A tienen un elemento minimal, i.e. un elemento m tal que

$$m \in Q \& \forall b \prec m. b \notin Q$$

Inducción bien fundada

- Las anteriores son casos particulares.

Relación bien fundada Es una relación binaria \prec sobre un conjunto A tal que no hay cadenas descendientes infinitas $\cdots \prec a_j \prec \cdots \prec a_1 \prec a_0$

Si $a \prec b$ decimos que a es un *predecesor* de b .

Es *irreflexiva*. $\nexists a/a \prec a$

$a \preceq b \iff a = b \text{ o } a \prec b$

Proposición Sea \prec una relación binaria en el conjunto A . La relación \prec está bien fundada iff todos los subconjuntos no vacíos Q de A tienen un elemento minimal, i.e. un elemento m tal que

$$m \in Q \& \forall b \prec m. b \notin Q$$

Demostración

- \prec está bien fundada $\Leftrightarrow m \in Q \& \forall b \prec m. b \notin Q$

Supongamos que todo conjunto no vacío de A tiene un elemento minimal. Si $\dots \prec a_j \prec \dots \prec a_1 \prec a_0$ es una cadena infinita descendente, entonces el conjunto $Q = \{a_i \mid i \in \omega\}$ sería no vacío sin un elemento minimal. Por lo tanto \prec está bien fundada.

- \prec está bien fundada $\Rightarrow m \in Q \& \forall b \prec m. b \notin Q$

Suponer que Q es un subconjunto no vacío de A . Construimos una cadena de elementos de esta forma: tomamos un elemento a_0 de Q . Inductivamente, asumimos que hemos construido en Q una cadena de elementos $a_n \prec \dots \prec a_0$. Si no existe un $b \in Q$ t.q. $b \prec a_n$ para la construcción. En caso contrario tomar $a_{n+1} = b$. Como \prec es bien fundada la cadena $\dots \prec a_j \prec \dots \prec a_1 \prec \dots \prec a_0$ no puede ser infinita. Si es finita, de la forma $a_n \prec \dots \prec a_0$ con $\forall b \prec a_n. b \notin Q$. Tomar como elemento minimal a_n .

El principio de Inducción bien fundada

- Sea \prec una relación bien fundada en un conjunto A . Sea P una propiedad. Entonces $\forall a \in A. P(a)$ iff:

$$\forall a \in A. ([\forall b \prec a. P(b)] \Rightarrow P(a))$$

- Prueba

La prueba se basa en la observación de que cualquier subconjunto no vacío Q de un conjunto A con una relación bien fundada \prec tiene un elemento minimal.

- \Rightarrow

Trivial

- \Leftarrow

Asumimos $\forall a \in A. ([\forall b \prec a. P(b)] \Rightarrow P(a))$ y producimos una contradicción suponiendo que $\neg P(a)$ para algún $a \in A$. Entonces, tiene que existir un elemento minimal m para el conjunto $\{a \in A \mid \neg P(a)\}$. Pero entonces $\neg P(m)$ y sin embargo $\forall b \prec m. P(b)$, lo cual contradice la asunción.

El principio de Inducción bien fundada

- Sea \prec una relación bien fundada en un conjunto A . Sea P una propiedad. Entonces $\forall a \in A. P(a)$ iff:

$$\forall a \in A. ([\forall b \prec a. P(b)] \Rightarrow P(a))$$

- Prueba

La prueba se basa en la observación de que cualquier subconjunto no vacío Q de un conjunto A con una relación bien fundada \prec tiene un elemento minimal.

- \Rightarrow

Trivial

- \Leftarrow

Asumimos $\forall a \in A. ([\forall b \prec a. P(b)] \Rightarrow P(a))$ y producimos una contradicción suponiendo que $\neg P(a)$ para algún $a \in A$. Entonces, tiene que existir un elemento minimal m para el conjunto $\{a \in A \mid \neg P(a)\}$. Pero entonces $\neg P(m)$ y sin embargo $\forall b \prec m. P(b)$, lo cual contradice la asunción.

El principio de Inducción bien fundada

- Sea \prec una relación bien fundada en un conjunto A . Sea P una propiedad. Entonces $\forall a \in A. P(a)$ iff:

$$\forall a \in A. ([\forall b \prec a. P(b)] \Rightarrow P(a))$$

- Prueba

La prueba se basa en la observación de que cualquier subconjunto no vacío Q de un conjunto A con una relación bien fundada \prec tiene un elemento minimal.

- \Rightarrow

Trivial

- \Leftarrow

Asumimos $\forall a \in A. ([\forall b \prec a. P(b)] \Rightarrow P(a))$ y producimos una contradicción suponiendo que $\neg P(a)$ para algún $a \in A$. Entonces, tiene que existir un elemento minimal m para el conjunto $\{a \in A \mid \neg P(a)\}$. Pero entonces $\neg P(m)$ y sin embargo $\forall b \prec m. P(b)$, lo cual contradice la asunción.

El principio de Inducción bien fundada

- Sea \prec una relación bien fundada en un conjunto A . Sea P una propiedad. Entonces $\forall a \in A. P(a)$ iff:

$$\forall a \in A. ([\forall b \prec a. P(b)] \Rightarrow P(a))$$

- Prueba

La prueba se basa en la observación de que cualquier subconjunto no vacío Q de un conjunto A con una relación bien fundada \prec tiene un elemento minimal.

- \Rightarrow

Trivial

- \Leftarrow

Asumimos $\forall a \in A. ([\forall b \prec a. P(b)] \Rightarrow P(a))$ y producimos una contradicción suponiendo que $\neg P(a)$ para algún $a \in A$. Entonces, tiene que existir un elemento minimal m para el conjunto $\{a \in A \mid \neg P(a)\}$. Pero entonces $\neg P(m)$ y sin embargo $\forall b \prec m. P(b)$, lo cual contradice la asunción.

- Inducción matemática

$$n \prec m \iff m = n + 1$$

- Inducción estructural

$a \prec b \iff a$ es una subexpresión directa de b

- Inducción matemática

$$n \prec m \iff m = n + 1$$

- Inducción estructural

$a \prec b \iff a$ es una subexpresión directa de b

- Algoritmo de Euclides (máximo común divisor)

Euclid \equiv

```
while (not (M = N)) do
  if (M <= N) then
    N := N - M;
  else
    M := M - N;
  fi
done
```

- Teorema (Terminación). Para todos los estados σ

$$\sigma(M) \geq 1 \ \& \ \sigma(N) \geq 1 \Rightarrow \exists \sigma'. \langle \textit{Euclid}, \sigma \rangle \rightarrow \sigma'$$

- Algoritmo de Euclides (máximo común divisor)

Euclid \equiv

```
while (not (M = N)) do
  if (M <= N) then
    N := N - M;
  else
    M := M - N;
  fi
done
```

- Teorema (Terminación). Para todos los estados σ

$$\sigma(M) \geq 1 \ \& \ \sigma(N) \geq 1 \Rightarrow \exists \sigma'. \langle \textit{Euclid}, \sigma \rangle \rightarrow \sigma'$$

- Deseamos probar que la propiedad

$$P(\sigma) \Leftrightarrow \exists \sigma'. \langle \text{Euclid}, \sigma \rangle \rightarrow \sigma'$$

se cumple $\forall \sigma \in \mathbf{S} = \{\sigma \in \Sigma \mid \sigma(M) \geq 1 \ \& \ \sigma(N) \geq 1\}$

- Usaremos inducción bien fundada sobre la relación:

$$\sigma' \prec \sigma \quad \text{iff} \quad (\sigma'(M) \leq \sigma(M) \ \& \ \sigma'(N) \leq \sigma(N)) \ \& \\ (\sigma'(M) \neq \sigma(M) \ \text{o} \ \sigma'(N) \neq \sigma(N))$$

- Deseamos probar que la propiedad

$$P(\sigma) \Leftrightarrow \exists \sigma'. \langle \text{Euclid}, \sigma \rangle \rightarrow \sigma'$$

se cumple $\forall \sigma \in \mathcal{S} = \{\sigma \in \Sigma \mid \sigma(M) \geq 1 \ \& \ \sigma(N) \geq 1\}$

- Usaremos inducción bien fundada sobre la relación:

$$\sigma' \prec \sigma \quad \text{iff} \quad (\sigma'(M) \leq \sigma(M) \ \& \ \sigma'(N) \leq \sigma(N)) \ \& \\ (\sigma'(M) \neq \sigma(M) \ \text{o} \ \sigma'(N) \neq \sigma(N))$$