Tecnología de la Programación

Semántica Operacional

David Cabrero Souto

Facultad de Informática
Universidade da Coruña

Curso 2006/2007

Verificación formal

- 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.

Verificación formal

- 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.

Verificación formal

- 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.

Verificación formal

- 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.

Verificación formal

- 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.

Verificación formal

- 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.

Verificación formal

- 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.

Lenguaje de ejemplo

- 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.

IMP- Definición (I)

Conjuntos sintácticos asociados con IMP.

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

IMP- Definición (I)

Conjuntos sintácticos asociados con IMP.

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

IMP- Definición (I)

Conjuntos sintácticos asociados con IMP.

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

IMP- Definición (I)

Conjuntos sintácticos asociados con IMP.

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

IMP- Definición (I)

Conjuntos sintácticos asociados con IMP.

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

IMP- Definición (I)

Conjuntos sintácticos asociados con IMP.

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

IMP- Definición (II)

Convenciones

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

IMP- Definición (II)

Convenciones

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

IMP- Definición (II)

Convenciones

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

IMP- Definición (II)

Convenciones

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

IMP- Definición (II)

Convenciones

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

IMP- Definición (*Aexp*)

- Expresiones aritméticas

$$
\begin{array}{lll}
\mathit{Aexp} & ::= & n \\
& | & X \\
& | & a_0 + a_1 \\
& | & a_0 - a_1 \\
& | & a_0 * a_1
\end{array}
$$

IMP- Definición (*Bexp*)

- Expresiones booleanas

$$
\begin{array}{lcl}
\mathit{Bexp} & ::= & \textbf{true} \\
& | & \textbf{false} \\
& | & a_0 = a_1 \\
& | & a_0 <= a_1 \\
& | & \textbf{not } b \\
& | & b_0 \textbf{ and } b_1 \\
& | & b_0 \textbf{ or } b_1
\end{array}
$$

IMP- Definición (*Comm*)

- Instrucciones

$$
\begin{array}{lcl}
\mathit{Comm} & ::= & \textbf{skip} \\
& | & X := a \\
& | & c_0 ; c_1 \\
& | & \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1 \textbf{ fi} \\
& | & \textbf{while } b \textbf{ do } c \textbf{ done}
\end{array}
$$

Estados

- El conjunto de estados Σ está formado por las funciones:
$$\sigma : \mathbf{Loc} -> \mathbf{N}$$
(Sólo variables enteras)
- Dado un estado σ, representaremos el valor de la posicion 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

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

Secuentes

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

$$\frac{< \textit{premisas} >}{< \textit{conclusion} >}$$

- $\dfrac{F_1 \ldots F_n}{G_1 \ldots G_m}$ $\qquad F_1 \ldots F_n \rightarrow G_1 \ldots G_m$
- Las usaremos para definir la semántica operacional.

Evaluación de expresiones aritméticas

- Cconstantes

$$\frac{}{< n, \sigma > \rightarrow n}$$

- Variables

$$\frac{}{< X, \sigma > \rightarrow \sigma(X)}$$

- Suma

$$\frac{< a_0, \sigma > \rightarrow n \;\; < a_1, \sigma > \rightarrow m}{< a_0 + a_1, \sigma > \rightarrow n + m}$$

- Resta

$$\frac{< a_0, \sigma > \rightarrow n \;\; < a_1, \sigma > \rightarrow m}{< a_0 - a_1, \sigma > \rightarrow n - m}$$

- División

$$\frac{< a_0, \sigma > \rightarrow n \;\; < a_1, \sigma > \rightarrow m}{< a_0 * a_1, \sigma > \rightarrow n * m}$$

Evaluación de expresiones aritméticas

- Cconstantes

$$\frac{}{< n, \sigma > \rightarrow n}$$

- Variables

$$\frac{}{< X, \sigma > \rightarrow \sigma(X)}$$

- Suma

$$\frac{< a_0, \sigma > \rightarrow n \quad < a_1, \sigma > \rightarrow m}{< a_0 + a_1, \sigma > \rightarrow n + m}$$

- Resta

$$\frac{< a_0, \sigma > \rightarrow n \quad < a_1, \sigma > \rightarrow m}{< a_0 - a_1, \sigma > \rightarrow n - m}$$

- División

$$\frac{< a_0, \sigma > \rightarrow n \quad < a_1, \sigma > \rightarrow m}{< a_0 * a_1, \sigma > \rightarrow n * m}$$

Evaluación de expresiones aritméticas

- Cconstantes

$$\frac{}{< n, \sigma > \rightarrow n}$$

- Variables

$$\frac{}{< X, \sigma > \rightarrow \sigma(X)}$$

- Suma

$$\frac{< a_0, \sigma > \rightarrow n \;\; < a_1, \sigma > \rightarrow m}{< a_0 + a_1, \sigma > \rightarrow n + m}$$

- Resta

$$\frac{< a_0, \sigma > \rightarrow n \;\; < a_1, \sigma > \rightarrow m}{< a_0 - a_1, \sigma > \rightarrow n - m}$$

- División

$$\frac{< a_0, \sigma > \rightarrow n \;\; < a_1, \sigma > \rightarrow m}{< a_0 * a_1, \sigma > \rightarrow n * m}$$

Evaluación de expresiones aritméticas

- Cconstantes

$$\frac{}{< n, \sigma > \rightarrow n}$$

- Variables

$$\frac{}{< X, \sigma > \rightarrow \sigma(X)}$$

- Suma

$$\frac{< a_0, \sigma > \rightarrow n \;\; < a_1, \sigma > \rightarrow m}{< a_0 + a_1, \sigma > \rightarrow n + m}$$

- Resta

$$\frac{< a_0, \sigma > \rightarrow n \;\; < a_1, \sigma > \rightarrow m}{< a_0 - a_1, \sigma > \rightarrow n - m}$$

- División

$$\frac{< a_0, \sigma > \rightarrow n \;\; < a_1, \sigma > \rightarrow m}{< a_0 * a_1, \sigma > \rightarrow n * m}$$

Evaluación de expresiones aritméticas

- Cconstantes

$$\frac{}{< n, \sigma > \rightarrow n}$$

- Variables

$$\frac{}{< X, \sigma > \rightarrow \sigma(X)}$$

- Suma

$$\frac{< a_0, \sigma > \rightarrow n \;\; < a_1, \sigma > \rightarrow m}{< a_0 + a_1, \sigma > \rightarrow n + m}$$

- Resta

$$\frac{< a_0, \sigma > \rightarrow n \;\; < a_1, \sigma > \rightarrow m}{< a_0 - a_1, \sigma > \rightarrow n - m}$$

- División

$$\frac{< a_0, \sigma > \rightarrow n \;\; < a_1, \sigma > \rightarrow m}{< a_0 * a_1, \sigma > \rightarrow n * m}$$

Evaluación de expresiones booleanas (I)

- Constantes

$$\frac{}{< \textbf{true}, \sigma > \rightarrow \textbf{true}} \qquad \frac{}{< \textbf{false}, \sigma > \rightarrow \textbf{false}}$$

- Comparación aritmética

$$\frac{< a_0, \sigma > \rightarrow n \;\; < a_1, \sigma > \rightarrow m}{< a_0 = a_1, \sigma > \rightarrow \textbf{true}} \quad n = m$$

$$\frac{< a_0, \sigma > \rightarrow n \;\; < a_1, \sigma > \rightarrow m}{< a_0 = a_1, \sigma > \rightarrow \textbf{false}} \quad n \neq m$$

$$\frac{< a_0, \sigma > \rightarrow n \;\; < a_1, \sigma > \rightarrow m}{< a_0 <= a_1, \sigma > \rightarrow \textbf{true}} \quad n <= m$$

$$\frac{< a_0, \sigma > \rightarrow n \;\; < a_1, \sigma > \rightarrow m}{< a_0 <= a_1, \sigma > \rightarrow \textbf{false}} \quad n > m$$

Evaluación de expresiones booleanas (I)

- Constantes

$$\frac{}{< \textbf{true}, \sigma > \rightarrow \textbf{true}} \qquad \frac{}{< \textbf{false}, \sigma > \rightarrow \textbf{false}}$$

- Comparación aritmética

$$\frac{< a_0, \sigma > \rightarrow n \;\; < a_1, \sigma > \rightarrow m}{< a_0 = a_1, \sigma > \rightarrow \textbf{true}} \quad n = m$$

$$\frac{< a_0, \sigma > \rightarrow n \;\; < a_1, \sigma > \rightarrow m}{< a_0 = a_1, \sigma > \rightarrow \textbf{false}} \quad n \neq m$$

$$\frac{< a_0, \sigma > \rightarrow n \;\; < a_1, \sigma > \rightarrow m}{< a_0 <= a_1, \sigma > \rightarrow \textbf{true}} \quad n <= m$$

$$\frac{< a_0, \sigma > \rightarrow n \;\; < a_1, \sigma > \rightarrow m}{< a_0 <= a_1, \sigma > \rightarrow \textbf{false}} \quad n > m$$

Evaluación de expresiones booleanas (II)

- Negación

$$\frac{< b, \sigma > \rightarrow \textbf{true}}{< \textbf{not } b, \sigma > \rightarrow \textbf{false}} \qquad \frac{< b, \sigma > \rightarrow \textbf{false}}{< \textbf{not } b, \sigma > \rightarrow \textbf{true}}$$

- Conectivas lógicas

$$\frac{< b_0, \sigma > \rightarrow t_0 \quad < b_1, \sigma > \rightarrow t_1}{< b_0 \textbf{ and } b_1, \sigma > \rightarrow t} \quad t = t_0 \wedge t_1$$

$$\frac{< b_0, \sigma > \rightarrow t_0 \quad < b_1, \sigma > \rightarrow t_1}{< b_0 \textbf{ or } b_1, \sigma > \rightarrow t} \quad t = t_0 \vee t_1$$

Evaluación de expresiones booleanas (II)

- Negación

$$\frac{< b, \sigma > \rightarrow \textbf{true}}{< \textbf{not } b, \sigma > \rightarrow \textbf{false}} \qquad \frac{< b, \sigma > \rightarrow \textbf{false}}{< \textbf{not } b, \sigma > \rightarrow \textbf{true}}$$

- Conectivas lógicas

$$\frac{< b_0, \sigma > \rightarrow t_0 \quad < b_1, \sigma > \rightarrow t_1}{< b_0 \textbf{ and } b_1, \sigma > \rightarrow t} \qquad t = t_0 \wedge t_1$$

$$\frac{< b_0, \sigma > \rightarrow t_0 \quad < b_1, \sigma > \rightarrow t_1}{< b_0 \textbf{ or } b_1, \sigma > \rightarrow t} \qquad t = t_0 \vee t_1$$

Evaluación de **and** optimizada

- Implementación habitual

$$\frac{< b_0, \sigma > \rightarrow \textbf{false}}{< b_0 \textbf{ and } b_1, \sigma > \rightarrow \textbf{false}}$$

$$\frac{< b_0, \sigma > \rightarrow \textbf{true} \quad < b_1, \sigma > \rightarrow \textbf{false}}{< b_0 \textbf{ and } b_1, \sigma > \rightarrow \textbf{false}}$$

$$\frac{< b_0, \sigma > \rightarrow \textbf{true} \quad < b_1, \sigma > \rightarrow \textbf{true}}{< b_0 \textbf{ and } b_1, \sigma > \rightarrow \textbf{true}}$$

Ejecución de instrucciones (I)

- Skip

$$\frac{}{< \textbf{skip}, \sigma > \rightarrow \sigma}$$

- Asignación

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

- Secuencia

$$\frac{< c_0, \sigma > \rightarrow \sigma'' \quad < c_1, \sigma'' > \rightarrow \sigma'}{< c_0; c_1, \sigma > \rightarrow \sigma'}$$

Ejecución de instrucciones (I)

- Skip

$$\frac{}{< \textbf{skip}, \sigma > \rightarrow \sigma}$$

- Asignación

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

- Secuencia

$$\frac{< c_0, \sigma > \rightarrow \sigma'' \quad < c_1, \sigma'' > \rightarrow \sigma'}{< c_0; c_1, \sigma > \rightarrow \sigma'}$$

Ejecución de instrucciones (I)

- Skip

$$\frac{}{< \textbf{skip}, \sigma > \rightarrow \sigma}$$

- Asignación

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

- Secuencia

$$\frac{< c_0, \sigma > \rightarrow \sigma'' \quad < c_1, \sigma'' > \rightarrow \sigma'}{< c_0; c_1, \sigma > \rightarrow \sigma'}$$

Ejecución de instrucciones (II)

- If

$$\frac{< b, \sigma > \rightarrow \texttt{true} \quad < c_0, \sigma > \rightarrow \sigma'}{< \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1 \textbf{ fi}, \sigma > \rightarrow \sigma'}$$

$$\frac{< b, \sigma > \rightarrow \texttt{false} \quad < c_1, \sigma > \rightarrow \sigma'}{< \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1 \textbf{ fi}, \sigma > \rightarrow \sigma'}$$

- While

$$\frac{< b, \sigma > \rightarrow \textbf{false}}{< \textbf{while } b \textbf{ do } c \textbf{ done}, \sigma > \rightarrow \sigma}$$

$$\frac{< b, \sigma > \rightarrow \textbf{true} \quad < c, \sigma > \rightarrow \sigma'' \quad < \textbf{while } b \textbf{ do } c \textbf{ done}, \sigma'' > \rightarrow \sigma'}{< \textbf{while } b \textbf{ do } c \textbf{ done}, \sigma > \rightarrow \sigma'}$$

Ejecución de instrucciones (II)

- If

$$\frac{< b, \sigma > \rightarrow \texttt{true} \quad < c_0, \sigma > \rightarrow \sigma'}{< \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1 \textbf{ fi}, \sigma > \rightarrow \sigma'}$$

$$\frac{< b, \sigma > \rightarrow \texttt{false} \quad < c_1, \sigma > \rightarrow \sigma'}{< \textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1 \textbf{ fi}, \sigma > \rightarrow \sigma'}$$

- While

$$\frac{< b, \sigma > \rightarrow \textbf{false}}{< \textbf{while } b \textbf{ do } c \textbf{ done}, \sigma > \rightarrow \sigma}$$

$$\frac{< b, \sigma > \rightarrow \textbf{true} \quad < c, \sigma > \rightarrow \sigma'' \quad < \textbf{while } b \textbf{ do } c \textbf{ done}, \sigma'' > \rightarrow \sigma'}{< \textbf{while } b \textbf{ do } c \textbf{ done}, \sigma > \rightarrow \sigma'}$$

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 $\sigma_0 = \{X \leftarrow 0\}$
  ¿$< a, \sigma_0 >$?

$$\dfrac{\dfrac{\dfrac{}{< X, \sigma_0 > \rightarrow 0} \quad \dfrac{}{< 5, \sigma_0 > \rightarrow 5}}{< (X + 5), \sigma_0 > \rightarrow 5} \quad \dfrac{\dfrac{}{< 7, \sigma_0 > \rightarrow 7} \quad \dfrac{}{< 9, \sigma_0 > \rightarrow 9}}{< (7 + 9), \sigma_0 > \rightarrow 16}}{< (X + 5) + (7 + 9), \sigma_0 > \rightarrow 21 >}$$

Terminación

- Si $w \equiv$ `while true do skip`, ¿ $\forall \sigma, \exists \sigma' t.q. < w, sigma > \rightarrow \sigma'$?

Preguntas

- ¿ Son iguales ?

```
skip
```

$\sim$

```
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**
  $(\forall n \in \mathcal{N}, \forall \sigma \in \Sigma, \quad < a_0, \sigma > \rightarrow n \Longleftrightarrow < a_1, \sigma > \rightarrow n)$
- $b_0 \sim b_1$ **iff**
  $(\forall t \forall \sigma \in \Sigma, \quad < b_0, \sigma > \rightarrow t \Longleftrightarrow < b_1, \sigma > \rightarrow t)$
- $c_0 \sim c_1$ **iff**
  $(\forall \sigma, \sigma' \in \Sigma, \quad < c_0, \sigma > \rightarrow \sigma' \Longleftrightarrow < c_1, \sigma > \rightarrow \sigma')$

Prueba simple

- Proposición
  Sea $w \equiv$ **while** b **do** c **done** con $b \in$ **Bexp**, $c \in$ **Comm** entonces:

  $w \sim$ **if** b **then** c; w **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.

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

Inducción estructural

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

$$< a, \sigma > \rightarrow m \quad \& \quad < a, \sigma > \rightarrow m' \Rightarrow m = m'$$

- Sea *P(a)* una propiedad de las expresiones aritméticas. Para demostrar que *(P(a)* se cumple para todas las expresiones artiméticas 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 $P(a_0 \times a_1)$.

Inducción estructural

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

$$< a, \sigma > \rightarrow m \quad \& \quad < a, \sigma > \rightarrow m' \Rightarrow m = m'$$

- Sea *P(a)* una propiedad de las expresiones aritméticas. Para demostrar que *(P(a)* se cumple para todas las expresiones artiméticas 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 $P(a_0 \times a_1)$.

Inducción estructural cont.

$$
\begin{array}{ll}
 & (\forall m \in N.P(m)) \\
\& & (\forall X \in Loc.P(X)) \\
\& & (\forall a_0, a_1 \in Aexp.P(a_0) \text{ y } P(a_1) \Rightarrow P(a_0 + a_1)) \\
\& & (\forall a_0, a_1 \in Aexp.P(a_0) \text{ y } P(a_1) \Rightarrow P(a_0 - a_1)) \\
\& & (\forall a_0, a_1 \in Aexp.P(a_0) \text{ y } P(a_1) \Rightarrow P(a_0 * a_1)) \\
\Rightarrow & \forall a \in Aexp.P(a)
\end{array}
$$

Inducción estructural cont.

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

$$< a, \sigma > \rightarrow m \quad \& \quad < a, \sigma > \rightarrow m' \Rightarrow m = m'$$

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 $\cdots \prec a_i \prec \cdots \prec a_1 \prec a_0$

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

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

$$a \preceq b \iff a = b \quad o \quad \prec b$$

Proposición Sea $\prec$ una relación binaria en el conjunto A. La relacion $\prec$ 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 $\cdots \prec a_i \prec \cdots \prec a_1 \prec a_0$

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

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

$$a \preceq b \iff a = b \quad o \quad < \prec b$$

Proposición Sea $\prec$ una relación binaria en el conjunto A. La relacion $\prec$ 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 $\cdots \prec a_i \prec \cdots \prec a_1 \prec a_0$

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

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

$$a \preceq b \iff a = b \quad o \quad \prec b$$

Proposición Sea $\prec$ una relación binaria en el conjunto A. La relacion $\prec$ 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 (cont.)

Demostración

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

  Supongamos que todo conjunto no vacio de A tiene un elemento minimal. Si $... \prec a_i \prec ... \prec a_1 \prec a_0$ es una cadena infinita descendente, entonces el conjunto $Q = \{a_i \mid i \in w\}$ 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 elememntos $a_n \prec ... \prec a_0$. Si no existe un $b \in Q$ t.q. $b \prec a_n$ parar la construcción. En caso contrario tomar $a_{n+1} = b$. Como $\prec$ es bien fundada la cadena $... \prec a_i \prec ... \prec a_1 ... \prec a_0$ no puede ser infinita. Si es finita, de la forma $a_n \prec ... \prec a_0$ con

El principio de Inducción bien fundada

- Sea $\prec$ una relacion 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 $\forall b \prec m.P(m)$, lo cual contradice la asumción.

El principio de Inducción bien fundada

- Sea $\prec$ una relacion 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 $\forall b \prec m.P(m)$, lo cual contradice la asumción.

El principio de Inducción bien fundada

- Sea $\prec$ una relacion 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 $\forall b \prec m.P(m)$, lo cual contradice la asumción.

El principio de Inducción bien fundada

- Sea $\prec$ una relacion 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 $\forall b \prec m.P(m)$, lo cual contradice la asumción.

Ejemplos

- Inducción matemática

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

- Inducción estructural

$$a \prec b \iff a \text{ es una subexpresión directa de } b$$

Ejemplos

- Inducción matemática

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

- Inducción estructural

$$a \prec b \iff a \text{ es una subexpresión directa de } b$$

Ejercicio

- Algoritmo de Euclides (máximo común divisor)

Euclid ≡

```
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

$$\sigma(M) \geq 1 \;\&\; \sigma(N) \geq 1 \Rightarrow \exists \sigma'. < \mathit{Euclid}, \sigma > \rightarrow \sigma'$$

Ejercicio

- 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

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

Ejercicio. Prueba

- Deseamos probar que la propiedad

$$P(\sigma) \Leftrightarrow \exists \sigma'. < \mathit{Euclid}, \sigma > \rightarrow \sigma'$$

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

- Usaremos inducción bien fundada sobre la relación:

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

Ejercicio. Prueba

- Deseamos probar que la propiedad

$$P(\sigma) \Leftrightarrow \exists \sigma'. < \mathit{Euclid}, \sigma > \rightarrow \sigma'$$

  se cumple $\forall \sigma \in S = \{\sigma \in \Sigma \mid \sigma(M) \geq 1 \ \& \ \sigma(N) \geq 1\}$
- Usaremos inducción bien fundada sobre la relación:

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