

# *Semánticas de lenguajes de programación*

## *¿Cómo se formulan los lenguajes de programación?*

Asignatura: Modelos de Programación II  
Proyecto Curricular de Ingeniería de Sistemas  
Universidad Distrital Francisco José de Caldas

2020 / Sesión sobre semánticas de lenguajes de programación



# CONTENIDOS

- 1 CONCEPTOS BÁSICOS
  - Sintaxis y semántica
- 2 REVISIÓN GENERAL A LOS TIPOS DE SEMÁNTICAS
- 3 EJEMPLO EXPLICATIVO PARA LAS SEMÁNTICAS
  - Semántica operacional
  - Semántica operacional estructural
  - Semántica operacional con estado
  - Semántica significativa
  - Semántica natural
  - Semántica axiomática



# CONTENIDOS

- 1 CONCEPTOS BÁSICOS
  - Sintaxis y semántica
- 2 REVISIÓN GENERAL A LOS TIPOS DE SEMÁNTICAS
- 3 EJEMPLO EXPLICATIVO PARA LAS SEMÁNTICAS
  - Semántica operacional
  - Semántica operacional estructural
  - Semántica operacional con estado
  - Semántica significativa
  - Semántica natural
  - Semántica axiomática



# CONTENIDOS

- 1 CONCEPTOS BÁSICOS
  - Sintaxis y semántica
- 2 REVISIÓN GENERAL A LOS TIPOS DE SEMÁNTICAS
- 3 EJEMPLO EXPLICATIVO PARA LAS SEMÁNTICAS
  - Semántica operacional
  - Semántica operacional estructural
  - Semántica operacional con estado
  - Semántica significativa
  - Semántica natural
  - Semántica axiomática





# SINTAXIS Y SUS BENEFICIOS

- La sintaxis se corresponde con la apariencia de buena formación del lenguaje.
- La sintaxis puede formalizarse por medio de una gramática o diagrama sintáctico.
- Los beneficios de la sintaxis, estilo BNF, aporta:
  - ① Estandarización de la sintaxis oficial del lenguaje.
  - ② Una guía para escribir programas sintácticamente correctos.
  - ③ Construcción de analizadores correctos para el compilador del lenguaje.
  - ④ Análisis formal de las propiedades del lenguaje; como determinar si el lenguaje es  $LL(k)$ ,  $LR(k)$  o ambiguo.
  - ⑤ La definición sintáctica puede usarse como entrada a compiladores o interpretadores.



# SINTAXIS Y SUS BENEFICIOS

- La sintaxis se corresponde con la apariencia de buena formación del lenguaje.
- La sintaxis puede formalizarse por medio de una gramática o diagrama sintáctico.
- Los beneficios de la sintaxis, estilo BNF, aporta:
  - ① Estandarización de la sintaxis oficial del lenguaje.
  - ② Una guía para escribir programas sintácticamente correctos.
  - ③ Construcción de analizadores correctos para el compilador del lenguaje.
  - ④ Análisis formal de las propiedades del lenguaje; como determinar si el lenguaje es  $LL(k)$ ,  $LR(k)$  o ambiguo.
  - ⑤ La definición sintáctica puede usarse como entrada a compiladores o interpretadores.



# SINTAXIS Y SUS BENEFICIOS

- La sintaxis se corresponde con la apariencia de buena formación del lenguaje.
- La sintaxis puede formalizarse por medio de una gramática o diagrama sintáctico.
- Los beneficios de la sintaxis, estilo BNF, aporta:
  - 1 Estandarización de la sintaxis oficial del lenguaje.
  - 2 Una guía para escribir programas sintácticamente correctos.
  - 3 Construcción de analizadores correctos para el compilador del lenguaje.
  - 4 Análisis formal de las propiedades del lenguaje; como determinar si el lenguaje es  $LL(k)$ ,  $LR(k)$  o ambiguo.
  - 5 La definición sintáctica puede usarse como entrada a compiladores o interpretadores.





# SINTAXIS Y SUS BENEFICIOS

- La sintaxis se corresponde con la apariencia de buena formación del lenguaje.
- La sintaxis puede formalizarse por medio de una gramática o diagrama sintáctico.
- Los beneficios de la sintaxis, estilo BNF, aporta:
  - 1 Estandarización de la sintaxis oficial del lenguaje.
  - 2 Una guía para escribir programas sintácticamente correctos.
  - 3 Construcción de analizadores correctos para el compilador del lenguaje.
  - 4 Análisis formal de las propiedades del lenguaje; como determinar si el lenguaje es  $LL(k)$ ,  $LR(k)$  o ambiguo.
  - 5 La definición sintáctica puede usarse como entrada a compiladores o interpretadores.



## SEMÁNTICA Y SUS BENEFICIOS

- La semántica se corresponde con el significado de los programas.
- La semántica también puede formalizarse y ser parte de la especificación de los lenguajes.
- Los beneficios de la definición semántica de los lenguajes de programación son:
  - ① Aportar una guía para que los usuarios comprendan los programas que escriben.
  - ② Servir de guía para escribir generadores de código correctos por parte de los implementadores.
  - ③ Permitir el análisis formal de propiedades de los lenguajes.
  - ④ Su uso como entrada a herramientas de generación de compiladores.



# SEMÁNTICA Y SUS BENEFICIOS

- La semántica se corresponde con el significado de los programas.
- La semántica también puede formalizarse y ser parte de la especificación de los lenguajes.
- Los beneficios de la definición semántica de los lenguajes de programación son:
  - ① Aportar una guía para que los usuarios comprendan los programas que escriben.
  - ② Servir de guía para escribir generadores de código correctos por parte de los implementadores.
  - ③ Permitir el análisis formal de propiedades de los lenguajes.
  - ④ Su uso como entrada a herramientas de generación de compiladores.







# PRINCIPALES MÉTODOS DE DEFINICIÓN SEMÁNTICA

- 1 **Semántica operacional.**
- 2 Semántica significativa.
- 3 Semántica axiomática.



# PRINCIPALES MÉTODOS DE DEFINICIÓN SEMÁNTICA

- 1 **Semántica operacional.**
- 2 **Semántica significativa.**
- 3 **Semántica axiomática.**



# PRINCIPALES MÉTODOS DE DEFINICIÓN SEMÁNTICA

- 1 **Semántica operacional.**
- 2 **Semántica significativa.**
- 3 **Semántica axiomática.**





## UN LENGUAJE ARITMÉTICO BÁSICO. SINTAXIS

Para comprender los métodos de definición semántica se usará el siguiente lenguaje:

$$E ::= N \mid E_1 + E_2$$

donde  $N$  representa el conjunto de números:  $\{0, 1, 2, \dots\}$



# CONTENIDOS

- 1 CONCEPTOS BÁSICOS
  - Sintaxis y semántica
- 2 REVISIÓN GENERAL A LOS TIPOS DE SEMÁNTICAS
- 3 EJEMPLO EXPLICATIVO PARA LAS SEMÁNTICAS
  - Semántica operacional
  - Semántica operacional estructural
  - Semántica operacional con estado
  - Semántica significativa
  - Semántica natural
  - Semántica axiomática





# CONTENIDOS

- 1 CONCEPTOS BÁSICOS
  - Sintaxis y semántica
- 2 REVISIÓN GENERAL A LOS TIPOS DE SEMÁNTICAS
- 3 **EJEMPLO EXPLICATIVO PARA LAS SEMÁNTICAS**
  - Semántica operacional
  - **Semántica operacional estructural**
  - Semántica operacional con estado
  - Semántica significativa
  - Semántica natural
  - Semántica axiomática







## PILA DE EJECUCIÓN Y PILA DE EXPRESIONES

Se usa el concepto de estado de ejecución como:

$$\langle s, c \rangle$$

Donde  $s$  es la pila de ejecución y  $c$  es una pila de expresiones aritméticas para este caso.

Una pila que contiene  $n$  ítems se representa como:

$$v_1 :: v_2 :: \dots :: v_n :: nil$$

El estado inicial para un programa  $p$  se escribe como:

$\langle nil, p :: nil \rangle$ ; la computación avanza hasta que aparece el estado  
 $\langle v :: nil, nil \rangle$ ; donde  $v$  es el resultado final.







# SEMÁNTICA DE TRANSICIÓN DE ESTADOS

Las reglas de re-escritura para una pequeña máquina abstracta de esta aritmética son:

- (1)  $\langle s, N :: c \rangle \Rightarrow \langle N :: s, c \rangle$
- (2)  $\langle s, (E_1 + E_2) :: c \rangle \Rightarrow \langle s, E_1 + E_2 :: c \rangle$
- (3)  $\langle s, E_1 + E_2 :: c \rangle \Rightarrow \langle s, E_1 :: E_2 :: + :: c \rangle$
- (4)  $\langle N_1 :: N_2 :: s, + :: c \rangle \Rightarrow \langle N' :: s, c \rangle$ ; donde  $N'$  es la suma de  $N_1$  y  $N_2$

- Se desea usar la semántica operacional definida arriba para computar  $(2 + 4) + (5 + 6)$ :

$$\begin{aligned} &< nil, (2+4)+(5+6) :: nil > \overset{3}{\Rightarrow} < nil, (2+4) :: (5+6) :: + :: nil > \overset{2}{\Rightarrow} < nil, 2+4 :: (5+6) :: + :: nil > \\ &\overset{3}{\Rightarrow} < nil, 2 :: 4 :: + :: (5+6) :: + :: nil > \overset{1}{\Rightarrow} < 2 :: nil, 4 :: + :: (5+6) :: + :: nil > \\ &\overset{1}{\Rightarrow} < 4 :: 2 :: nil, + :: (5+6) :: + :: nil > \overset{4}{\Rightarrow} < 6 :: nil, (5+6) :: + :: nil > \overset{2}{\Rightarrow} < 6 :: nil, 5+6 :: + :: nil > \\ &\overset{3}{\Rightarrow} < 6 :: nil, 5 :: 6 :: + :: + :: nil > \overset{1}{\Rightarrow} < 5 :: 6 :: nil, 6 :: + :: + :: nil > \overset{1}{\Rightarrow} < 6 :: 5 :: 6 :: nil, + :: + :: nil > \\ &\overset{4}{\Rightarrow} < 11 :: 6 :: nil, + :: nil > \overset{4}{\Rightarrow} < 17 :: nil, nil > \end{aligned}$$



# SEMÁNTICA DE TRANSICIÓN DE ESTADOS

Las reglas de re-escritura para una pequeña máquina abstracta de esta aritmética son:

- (1)  $\langle s, N :: c \rangle \Rightarrow \langle N :: s, c \rangle$
- (2)  $\langle s, (E_1 + E_2) :: c \rangle \Rightarrow \langle s, E_1 + E_2 :: c \rangle$
- (3)  $\langle s, E_1 + E_2 :: c \rangle \Rightarrow \langle s, E_1 :: E_2 :: + :: c \rangle$
- (4)  $\langle N_1 :: N_2 :: s, + :: c \rangle \Rightarrow \langle N' :: s, c \rangle$ ; donde  $N'$  es la suma de  $N_1$  y  $N_2$

- Se desea usar la semántica operacional definida arriba para computar  $(2 + 4) + (5 + 6)$ :

$$\begin{aligned} &< nil, (2+4)+(5+6) :: nil > \stackrel{3}{\Rightarrow} < nil, (2+4) :: (5+6) :: + :: nil > \stackrel{2}{\Rightarrow} < nil, 2+4 :: (5+6) :: + :: nil > \\ &\stackrel{3}{\Rightarrow} < nil, 2 :: 4 :: + :: (5+6) :: + :: nil > \stackrel{1}{\Rightarrow} < 2 :: nil, 4 :: + :: (5+6) :: + :: nil > \\ &\stackrel{1}{\Rightarrow} < 4 :: 2 :: nil, + :: (5+6) :: + :: nil > \stackrel{4}{\Rightarrow} < 6 :: nil, (5+6) :: + :: nil > \stackrel{2}{\Rightarrow} < 6 :: nil, 5+6 :: + :: nil > \\ &\stackrel{3}{\Rightarrow} < 6 :: nil, 5 :: 6 :: + :: + :: nil > \stackrel{1}{\Rightarrow} < 5 :: 6 :: nil, 6 :: + :: + :: nil > \stackrel{1}{\Rightarrow} < 6 :: 5 :: 6 :: nil, + :: + :: nil > \\ &\stackrel{4}{\Rightarrow} < 11 :: 6 :: nil, + :: nil > \stackrel{4}{\Rightarrow} < 17 :: nil, nil > \end{aligned}$$



# SEMÁNTICA DE TRANSICIÓN DE ESTADOS

Las reglas de re-escritura para una pequeña máquina abstracta de esta aritmética son:

- (1)  $\langle s, N :: c \rangle \Rightarrow \langle N :: s, c \rangle$
- (2)  $\langle s, (E_1 + E_2) :: c \rangle \Rightarrow \langle s, E_1 + E_2 :: c \rangle$
- (3)  $\langle s, E_1 + E_2 :: c \rangle \Rightarrow \langle s, E_1 :: E_2 :: + :: c \rangle$
- (4)  $\langle N_1 :: N_2 :: s, + :: c \rangle \Rightarrow \langle N' :: s, c \rangle$ ; donde  $N'$  es la suma de  $N_1$  y  $N_2$

- Se desea usar la semántica operacional definida arriba para computar  $(2 + 4) + (5 + 6)$ :

$$\begin{aligned} &< nil, (2+4)+(5+6) :: nil > \stackrel{3}{\Rightarrow} < nil, (2+4) :: (5+6) :: + :: nil > \stackrel{2}{\Rightarrow} < nil, 2+4 :: (5+6) :: + :: nil > \\ &\stackrel{3}{\Rightarrow} < nil, 2 :: 4 :: + :: (5+6) :: + :: nil > \stackrel{1}{\Rightarrow} < 2 :: nil, 4 :: + :: (5+6) :: + :: nil > \\ &\stackrel{1}{\Rightarrow} < 4 :: 2 :: nil, + :: (5+6) :: + :: nil > \stackrel{4}{\Rightarrow} < 6 :: nil, (5+6) :: + :: nil > \stackrel{2}{\Rightarrow} < 6 :: nil, 5+6 :: + :: nil > \\ &\stackrel{3}{\Rightarrow} < 6 :: nil, 5 :: 6 :: + :: + :: nil > \stackrel{1}{\Rightarrow} < 5 :: 6 :: nil, 6 :: + :: + :: nil > \stackrel{1}{\Rightarrow} < 6 :: 5 :: 6 :: nil, + :: + :: nil > \\ &\stackrel{4}{\Rightarrow} < 11 :: 6 :: nil, + :: nil > \stackrel{4}{\Rightarrow} < 17 :: nil, nil > \end{aligned}$$



# SEMÁNTICA DE TRANSICIÓN DE ESTADOS

Las reglas de re-escritura para una pequeña máquina abstracta de esta aritmética son:

- (1)  $\langle s, N :: c \rangle \Rightarrow \langle N :: s, c \rangle$
- (2)  $\langle s, (E_1 + E_2) :: c \rangle \Rightarrow \langle s, E_1 + E_2 :: c \rangle$
- (3)  $\langle s, E_1 + E_2 :: c \rangle \Rightarrow \langle s, E_1 :: E_2 :: + :: c \rangle$
- (4)  $\langle N_1 :: N_2 :: s, + :: c \rangle \Rightarrow \langle N' :: s, c \rangle$ ; donde  $N'$  es la suma de  $N_1$  y  $N_2$

- Se desea usar la semántica operacional definida arriba para computar  $(2 + 4) + (5 + 6)$ :

$$\begin{aligned} &< nil, (2+4)+(5+6) :: nil > \stackrel{3}{\Rightarrow} < nil, (2+4) :: (5+6) :: + :: nil > \stackrel{2}{\Rightarrow} < nil, 2+4 :: (5+6) :: + :: nil > \\ &\stackrel{3}{\Rightarrow} < nil, 2 :: 4 :: + :: (5+6) :: + :: nil > \stackrel{1}{\Rightarrow} < 2 :: nil, 4 :: + :: (5+6) :: + :: nil > \\ &\stackrel{1}{\Rightarrow} < 4 :: 2 :: nil, + :: (5+6) :: + :: nil > \stackrel{4}{\Rightarrow} < 6 :: nil, (5+6) :: + :: nil > \stackrel{2}{\Rightarrow} < 6 :: nil, 5+6 :: + :: nil > \\ &\stackrel{3}{\Rightarrow} < 6 :: nil, 5 :: 6 :: + :: + :: nil > \stackrel{1}{\Rightarrow} < 5 :: 6 :: nil, 6 :: + :: + :: nil > \stackrel{1}{\Rightarrow} < 6 :: 5 :: 6 :: nil, + :: + :: \\ &nil > \\ &\stackrel{4}{\Rightarrow} < 11 :: 6 :: nil, + :: nil > \stackrel{4}{\Rightarrow} < 17 :: nil, nil > \end{aligned}$$





# CONTENIDOS

- 1 CONCEPTOS BÁSICOS
  - Sintaxis y semántica
- 2 REVISIÓN GENERAL A LOS TIPOS DE SEMÁNTICAS
- 3 EJEMPLO EXPLICATIVO PARA LAS SEMÁNTICAS
  - Semántica operacional
  - Semántica operacional estructural
  - Semántica operacional con estado
  - **Semántica significativa**
  - Semántica natural
  - Semántica axiomática



## CONCEPTUALIZACIÓN BÁSICA

- Enfatiza en que un programa tiene un significado matemático subyacente que es independiente de cualquier estrategia de computación.
- La asignación de significados a programas se hace de manera *composicional*, es decir, el significado de una expresión se construye desde sus subexpresiones.
- Por ejemplo:  $Nat = \{0, 1, 2, \dots\}$  y

$$plus : Nat \times Nat \longrightarrow Nat$$



## CONCEPTUALIZACIÓN BÁSICA

- Enfatiza en que un programa tiene un significado matemático subyacente que es independiente de cualquier estrategia de computación.
- La asignación de significados a programas se hace de manera *composicional*, es decir, el significado de una expresión se construye desde sus subexpresiones.
- Por ejemplo:  $Nat = \{0, 1, 2, \dots\}$  y

$$plus : Nat \times Nat \longrightarrow Nat$$





## CONCEPTUALIZACIÓN BÁSICA

- Enfatiza en que un programa tiene un significado matemático subyacente que es independiente de cualquier estrategia de computación.
- La asignación de significados a programas se hace de manera *composicional*, es decir, el significado de una expresión se construye desde sus subexpresiones.
- Por ejemplo:  $Nat = \{0, 1, 2, \dots\}$  y

$$plus : Nat \times Nat \longrightarrow Nat$$



## CONCEPTUALIZACIÓN BÁSICA

- Enfatiza en que un programa tiene un significado matemático subyacente que es independiente de cualquier estrategia de computación.
- La asignación de significados a programas se hace de manera *composicional*, es decir, el significado de una expresión se construye desde sus subexpresiones.
- Por ejemplo:  $Nat = \{0, 1, 2, \dots\}$  y

$$plus : Nat \times Nat \longrightarrow Nat$$



## SEMÁNTICA SIGNIFICATIVA. EL EJEMPLO:

$$\begin{aligned}\mathcal{E} &: \text{Expression} \rightarrow \text{Nat} \\ \mathcal{E}[\![N]\!] &= N \\ \mathcal{E}[\![E_1 + E_2]\!] &= \text{plus}(\mathcal{E}[\![E_1]\!], \mathcal{E}[\![E_2]\!])\end{aligned}$$

Un ejemplo:

$$\begin{aligned}\mathcal{E}[\![ (1 + 2) + (4 + 5) ]\!] &= \text{plus}(\mathcal{E}[\![1 + 2]\!], \mathcal{E}[\![4 + 5]\!]) \\ &= \text{plus}(\text{plus}(\mathcal{E}[\![1]\!], \mathcal{E}[\![2]\!]), \text{plus}(\mathcal{E}[\![4]\!], \mathcal{E}[\![5]\!])) \\ &= \text{plus}(3, 9) = 12\end{aligned}$$

$$\begin{aligned}\mathcal{E}[\![ (1 + 2) + (4 + 5) ]\!] &= \text{plus}(\mathcal{E}[\![1 + 2]\!], \mathcal{E}[\![4 + 5]\!]) \\ \mathcal{E}[\![1 + 2]\!] &= \text{plus}(\mathcal{E}[\![1]\!], \mathcal{E}[\![2]\!]) \\ \mathcal{E}[\![4 + 5]\!] &= \text{plus}(\mathcal{E}[\![4]\!], \mathcal{E}[\![5]\!]) \\ \mathcal{E}[\![1]\!] &= 1 & \mathcal{E}[\![2]\!] &= 2 \\ \mathcal{E}[\![4]\!] &= 4 & \mathcal{E}[\![5]\!] &= 5\end{aligned}$$





## SEMÁNTICA NATURAL

- Es un método semántico híbrido que combina las ventajas de la semántica operacional y la semántica significativa
- Por lo anterior, combina reglas de inferencia con reglas de re-escritura.
- Por ejemplo, las reglas de la semántica natural para el ejemplo son:

$$\begin{array}{c} N \Downarrow N \\ \hline E_1 \Downarrow n_1 \quad E_2 \Downarrow n_2 \\ \hline E_1 + E_2 \Downarrow m \end{array}$$

donde  $m$  es la suma de  $n_1$  y  $n_2$ :

$$\begin{array}{c} 1 \Downarrow 1 \quad 2 \Downarrow 2 \quad 4 \Downarrow 4 \quad 5 \Downarrow 5 \\ \hline (1 + 2) \Downarrow 3 \quad (4 + 5) \Downarrow 9 \\ \hline (1 + 2) + (4 + 5) \Downarrow 12 \end{array}$$



# SEMÁNTICA NATURAL

- Es un método semántico híbrido que combina las ventajas de la semántica operacional y la semántica significativa
- Por lo anterior, combina reglas de inferencia con reglas de re-escritura.
- Por ejemplo, las reglas de la semántica natural para el ejemplo son:

$$\begin{array}{c} N \Downarrow N \\ \hline E_1 \Downarrow n_1 \quad E_2 \Downarrow n_2 \\ \hline E_1 + E_2 \Downarrow m \end{array}$$

$$\begin{array}{r} \frac{1 \Downarrow 1 \quad 2 \Downarrow 2}{(1+2) \Downarrow 3} \quad \frac{4 \Downarrow 4 \quad 5 \Downarrow 5}{(4+5) \Downarrow 9} \\ \hline (1+2) + (4+5) \Downarrow 12 \end{array}$$



# SEMÁNTICA NATURAL

- Es un método semántico híbrido que combina las ventajas de la semántica operacional y la semántica significativa
- Por lo anterior, combina reglas de inferencia con reglas de re-escritura.
- Por ejemplo, las reglas de la semántica natural para el ejemplo son:

$$\frac{E_1 \Downarrow n_1 \quad E_2 \Downarrow n_2}{E_1 + E_2 \Downarrow m}$$

donde  $m$  es la suma de  $n_1$  y  $n_2$ :

$$\begin{array}{r} \frac{1 \Downarrow 1 \quad 2 \Downarrow 2}{(1+2) \Downarrow 3} \quad \frac{4 \Downarrow 4 \quad 5 \Downarrow 5}{(4+5) \Downarrow 9} \\ \hline (1+2) + (4+5) \Downarrow 12 \end{array}$$



# SEMÁNTICA NATURAL

- Es un método semántico híbrido que combina las ventajas de la semántica operacional y la semántica significativa
- Por lo anterior, combina reglas de inferencia con reglas de re-escritura.
- Por ejemplo, las reglas de la semántica natural para el ejemplo son:

$$\begin{array}{c} N \Downarrow N \\ \hline E_1 \Downarrow n_1 \quad E_2 \Downarrow n_2 \\ \hline E_1 + E_2 \Downarrow m \end{array}$$

donde  $m$  es la suma de  $n_1$  y  $n_2$ :

$$\begin{array}{r} \frac{1 \Downarrow 1 \quad 2 \Downarrow 2}{(1+2) \Downarrow 3} \quad \frac{4 \Downarrow 4 \quad 5 \Downarrow 5}{(4+5) \Downarrow 9} \\ \hline (1+2) + (4+5) \Downarrow 12 \end{array}$$





# SEMÁNTICA NATURAL

- Es un método semántico híbrido que combina las ventajas de la semántica operacional y la semántica significativa
- Por lo anterior, combina reglas de inferencia con reglas de re-escritura.
- Por ejemplo, las reglas de la semántica natural para el ejemplo son:

$$\begin{array}{c} N \Downarrow N \\ \hline E_1 \Downarrow n_1 \quad E_2 \Downarrow n_2 \\ \hline E_1 + E_2 \Downarrow m \end{array}$$

donde  $m$  es la suma de  $n_1$  y  $n_2$ :

$$\begin{array}{r} \frac{1 \Downarrow 1 \quad 2 \Downarrow 2}{(1+2) \Downarrow 3} \quad \frac{4 \Downarrow 4 \quad 5 \Downarrow 5}{(4+5) \Downarrow 9} \\ \hline (1+2) + (4+5) \Downarrow 12 \end{array}$$



# CONTENIDOS

- 1 CONCEPTOS BÁSICOS
  - Sintaxis y semántica
- 2 REVISIÓN GENERAL A LOS TIPOS DE SEMÁNTICAS
- 3 EJEMPLO EXPLICATIVO PARA LAS SEMÁNTICAS
  - Semántica operacional
  - Semántica operacional estructural
  - Semántica operacional con estado
  - Semántica significativa
  - Semántica natural
  - Semántica axiomática



# SEMÁNTICA AXIOMÁTICA

$N : is\_even$  if  $N \bmod 2 = 0$        $N : is\_odd$  if  $N \bmod 2 = 1$

$$\frac{E_1 : p_1 \quad E_2 : p_2}{E_1 + E_2 : p_3} \quad \text{where } p_3 = \begin{cases} is\_even & \text{if } p_1 = p_2 \\ is\_odd & \text{otherwise} \end{cases}$$

$$\frac{\frac{1 : is\_odd \quad 2 : is\_even}{1 + 2 : is\_odd} \quad \frac{4 : is\_even \quad 5 : is\_odd}{4 + 5 : is\_odd}}{(1 + 2) + (4 + 5) : is\_even}$$





[Schmidt,2004] David A. Schmidt with Editor-in-Chief Allen B. Tucker

Programming Language Semantics

*Publicado por Chapman & Hall/CRC with ACM in Computer Science Handbook,2004.*

