Implementation of syntax and semantic of grammars through parser combinators and attribute grammars

John Jairo Silva
jsilvazu@gmail.com

12 de febrero de 2013

Orden

- Semánticas
- Métodos semánticos
- Parser combinadores
- Gramáticas de atributos
- Lenguaje While
- Máquina abstracta
- Traductor
- Problemas
- Conclusiones y trabajos futuros

Semántica formal

La semántica formal especifica el comportamiento o significado en un programa

Importa porque

- Puede mostrar ambigüedad
- Puede formar una base solida para el análisis y verificación de un programa

Sintaxis Evalúa la estructura gramatical de un programa

$$< z := x; x := y; y := z >$$

Semántica Evalúa el significado de un programa

Métodos semánticos

Semántica operacional El significado es visto desde la ejecución del programa en un máquina. En este punto solo importa el cálculo de **como** se está haciendo

Semántica denotacional El significado del programa es modelado por medio de objetos matemáticos que representan el efecto de ejecutar un programa. Solo importa el **efecto de ejecución**, no el como se obtiene.

Semántica Axiomática Lo que importa en esta semántica es que las propiedades del programa **sean parcialmente correctas**. Un programa es parcialmente correcto, con respecto a una condición previa y una condición posterior, si siempre que el estado inicial cumple la condición y el programa termina, es así que el estado final está garantizado para cumplir la condición posterior.

Semántica Operacional

Le importa el como se ejecuta un programa

Semántica estructural Small step

$$
\begin{aligned}
& \langle \mathtt{z{:=}x;\ x{:=}y;\ y{:=}z},\ [\mathtt{x}\mapsto\mathbf{5},\ \mathtt{y}\mapsto\mathbf{7},\ \mathtt{z}\mapsto\mathbf{0}]\rangle \\
\Rightarrow\ & \langle \mathtt{x{:=}y;\ y{:=}z},\ [\mathtt{x}\mapsto\mathbf{5},\ \mathtt{y}\mapsto\mathbf{7},\ \mathtt{z}\mapsto\mathbf{5}]\rangle \\
\Rightarrow\ & \langle \mathtt{y{:=}z},\ [\mathtt{x}\mapsto\mathbf{7},\ \mathtt{y}\mapsto\mathbf{7},\ \mathtt{z}\mapsto\mathbf{5}]\rangle \\
\Rightarrow\ & [\mathtt{x}\mapsto\mathbf{7},\ \mathtt{y}\mapsto\mathbf{5},\ \mathtt{z}\mapsto\mathbf{5}]
\end{aligned}
$$

Semántica natural Big step

$$
\frac{\langle \mathtt{z{:=}x},\ s_0\rangle \to s_1 \qquad \langle \mathtt{x{:=}y},\ s_1\rangle \to s_2}{\langle \mathtt{z{:=}x;\ x{:=}y},\ s_0\rangle \to s_2 \qquad \langle \mathtt{y{:=}z},\ s_2\rangle \to s_3}
$$

$$
\langle \mathtt{z{:=}x;\ x{:=}y;\ y{:=}z},\ s_0\rangle \to s_3
$$

Semántica Denotacional

Se miran los efectos de ejecutar un programa

$$\mathcal{S}[\![\mathrm{z{:=}x;\ x{:=}y;\ y{:=}z}]\!] = \mathcal{S}[\![\mathrm{y{:=}z}]\!] \circ \mathcal{S}[\![\mathrm{x{:=}y}]\!] \circ \mathcal{S}[\![\mathrm{z{:=}x}]\!]$$

Ejecución

$$\begin{aligned}
&\mathcal{S}[\![\mathrm{z{:=}x;\ x{:=}y;\ y{:=}z}]\!]([\mathrm{x}\mapsto\mathbf{5},\ \mathrm{y}\mapsto\mathbf{7},\ \mathrm{z}\mapsto\mathbf{0}]) \\
&\quad = (\mathcal{S}[\![\mathrm{y{:=}z}]\!] \circ \mathcal{S}[\![\mathrm{x{:=}y}]\!] \circ \mathcal{S}[\![\mathrm{z{:=}x}]\!])([\mathrm{x}\mapsto\mathbf{5},\ \mathrm{y}\mapsto\mathbf{7},\ \mathrm{z}\mapsto\mathbf{0}]) \\
&\quad = \mathcal{S}[\![\mathrm{y{:=}z}]\!](\mathcal{S}[\![\mathrm{x{:=}y}]\!](\mathcal{S}[\![\mathrm{z{:=}x}]\!]([\mathrm{x}\mapsto\mathbf{5},\ \mathrm{y}\mapsto\mathbf{7},\ \mathrm{z}\mapsto\mathbf{0}]))) \\
&\quad = \mathcal{S}[\![\mathrm{y{:=}z}]\!](\mathcal{S}[\![\mathrm{x{:=}y}]\!]([\mathrm{x}\mapsto\mathbf{5},\ \mathrm{y}\mapsto\mathbf{7},\ \mathrm{z}\mapsto\mathbf{5}])) \\
&\quad = \mathcal{S}[\![\mathrm{y{:=}z}]\!]([\mathrm{x}\mapsto\mathbf{7},\ \mathrm{y}\mapsto\mathbf{7},\ \mathrm{z}\mapsto\mathbf{5}]) \\
&\quad = [\mathrm{x}\mapsto\mathbf{7},\ \mathrm{y}\mapsto\mathbf{5},\ \mathrm{z}\mapsto\mathbf{5}]
\end{aligned}$$

Semántica Axiomática

Interesa las propiedades de corrección parcial de programas

$\{\ \mathtt{x{=}n} \wedge \mathtt{y{=}m}\ \}\ \mathtt{z{:=}x;\ x{:=}y;\ y{:=}z}\ \{\ \mathtt{y{=}n} \wedge \mathtt{x{=}m}\ \}$

Ejecución

$$\dfrac{\dfrac{\{\ p_0\ \}\ \mathtt{z{:=}x}\ \{\ p_1\ \} \qquad \{\ p_1\ \}\ \mathtt{x{:=}y}\ \{\ p_2\ \}}{\{\ p_0\ \}\ \mathtt{z{:=}x;\ x{:=}y}\ \{\ p_2\ \}} \qquad \{\ p_2\ \}\ \mathtt{y{:=}z}\ \{\ p_3\ \}}{\{\ p_0\ \}\ \mathtt{z{:=}x;\ x{:=}y;\ y{:=}z}\ \{\ p_3\ \}}$$

$$\begin{aligned} p_0 &= \mathtt{x{=}n} \wedge \mathtt{y{=}m} \\ p_1 &= \mathtt{z{=}n} \wedge \mathtt{y{=}m} \\ p_2 &= \mathtt{z{=}n} \wedge \mathtt{x{=}m} \\ p_3 &= \mathtt{y{=}n} \wedge \mathtt{x{=}m} \end{aligned}$$

Parser combinadores

Parser Programa que analiza un texto para determinar su estructura lógica, en un compilador toma un programa escrito en text y lo retorna como un AST que representa la estructura del programa
Los **parser combinadores** son funciones de orden para construir analizadores como especificaciones ejecutables de gramáticas. Se les llama combinadores porque al procesar un parser con otro, producen un nuevo parser

- Reconocen gramáticas ambiguas
- Backtracking

  <*> Composición secuencial

  <|> Composición alternativa

  <$> Cambio de tipo de resultado de un parser

Gramáticas de atributos

Las gramáticas de atributos permiten representar, aparte de la sintaxis de los lenguajes, la forma en la que las frases de dichos lenguajes serán procesadas

Lenguaje While

$$
\begin{array}{lcl}
a & ::= & n \mid x \mid a_1 + a_2 \mid a_1 \star a_2 \mid a_1 - a_2 \\
b & ::= & \texttt{true} \mid \texttt{false} \mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b \mid b_1 \wedge b_2 \\
S & ::= & x := a \mid \texttt{skip} \mid S_1 \; ; \; S_2 \mid \texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2 \\
 & \mid & \texttt{while } b \texttt{ do } S
\end{array}
$$

Semántica lenguaje **While**

Aritmética

$$
\begin{aligned}
\mathcal{A}[\![n]\!]s &= \mathcal{N}[\![n]\!] \\
\mathcal{A}[\![x]\!]s &= s\ x \\
\mathcal{A}[\![a_1 + a_2]\!]s &= \mathcal{A}[\![a_1]\!]s + \mathcal{A}[\![a_2]\!]s \\
\mathcal{A}[\![a_1 \star a_2]\!]s &= \mathcal{A}[\![a_1]\!]s \cdot \mathcal{A}[\![a_2]\!]s \\
\mathcal{A}[\![a_1 - a_2]\!]s &= \mathcal{A}[\![a_1]\!]s - \mathcal{A}[\![a_2]\!]s
\end{aligned}
$$

Booleana

$$
\begin{aligned}
\mathcal{B}[\![\texttt{true}]\!]s &= \mathbf{tt} \\
\mathcal{B}[\![\texttt{false}]\!]s &= \mathbf{ff} \\
\mathcal{B}[\![a_1 = a_2]\!]s &= \begin{cases} \mathbf{tt} & \text{if } \mathcal{A}[\![a_1]\!]s = \mathcal{A}[\![a_2]\!]s \\ \mathbf{ff} & \text{if } \mathcal{A}[\![a_1]\!]s \neq \mathcal{A}[\![a_2]\!]s \end{cases} \\
\mathcal{B}[\![a_1 \leq a_2]\!]s &= \begin{cases} \mathbf{tt} & \text{if } \mathcal{A}[\![a_1]\!]s \leq \mathcal{A}[\![a_2]\!]s \\ \mathbf{ff} & \text{if } \mathcal{A}[\![a_1]\!]s > \mathcal{A}[\![a_2]\!]s \end{cases} \\
\mathcal{B}[\![\neg\ b]\!]s &= \begin{cases} \mathbf{tt} & \text{if } \mathcal{B}[\![b]\!]s = \mathbf{ff} \\ \mathbf{ff} & \text{if } \mathcal{B}[\![b]\!]s = \mathbf{tt} \end{cases} \\
\mathcal{B}[\![b_1 \wedge b_2]\!]s &= \begin{cases} \mathbf{tt} & \text{if } \mathcal{B}[\![b_1]\!]s = \mathbf{tt} \text{ and } \mathcal{B}[\![b_2]\!]s = \mathbf{tt} \\ \mathbf{ff} & \text{if } \mathcal{B}[\![b_1]\!]s = \mathbf{ff} \text{ or } \mathcal{B}[\![b_2]\!]s = \mathbf{ff} \end{cases}
\end{aligned}
$$

Máquina abstracta

$$
\begin{array}{lll}
inst & ::= & \text{PUSH-}n \mid \text{ADD} \mid \text{MULT} \mid \text{SUB} \\
& \mid & \text{TRUE} \mid \text{FALSE} \mid \text{EQ} \mid \text{LE} \mid \text{AND} \mid \text{NEG} \\
& \mid & \text{FETCH-}x \mid \text{STORE-}x \\
& \mid & \text{NOOP} \mid \text{BRANCH}(c,\ c) \mid \text{LOOP}(c,\ c) \\
c & ::= & \varepsilon \mid inst{:}c
\end{array}
$$

Semántica máquina abstracta 1

$$
\begin{array}{lcll}
\langle \text{PUSH-}n{:}c,\ e,\ s\rangle & \triangleright & \langle c,\ \mathcal{N}[\![n]\!]{:}e,\ s\rangle & \\
\langle \text{ADD}{:}c,\ z_1{:}z_2{:}e,\ s\rangle & \triangleright & \langle c,\ (z_1{+}z_2){:}e,\ s\rangle & \text{if } z_1,\ z_2{\in}\mathbf{Z} \\
\langle \text{MULT}{:}c,\ z_1{:}z_2{:}e,\ s\rangle & \triangleright & \langle c,\ (z_1{\star}z_2){:}e,\ s\rangle & \text{if } z_1,\ z_2{\in}\mathbf{Z} \\
\langle \text{SUB}{:}c,\ z_1{:}z_2{:}e,\ s\rangle & \triangleright & \langle c,\ (z_1{-}z_2){:}e,\ s\rangle & \text{if } z_1,\ z_2{\in}\mathbf{Z} \\
\langle \text{TRUE}{:}c,\ e,\ s\rangle & \triangleright & \langle c,\ \mathbf{tt}{:}e,\ s\rangle & \\
\langle \text{FALSE}{:}c,\ e,\ s\rangle & \triangleright & \langle c,\ \mathbf{ff}{:}e,\ s\rangle & \\
\langle \text{EQ}{:}c,\ z_1{:}z_2{:}e,\ s\rangle & \triangleright & \langle c,\ (z_1{=}z_2){:}e,\ s\rangle & \text{if } z_1,\ z_2{\in}\mathbf{Z} \\
\langle \text{LE}{:}c,\ z_1{:}z_2{:}e,\ s\rangle & \triangleright & \langle c,\ (z_1{\leq}z_2){:}e,\ s\rangle & \text{if } z_1,\ z_2{\in}\mathbf{Z} \\
\langle \text{AND}{:}c,\ t_1{:}t_2{:}e,\ s\rangle & \triangleright & & \\
\multicolumn{4}{c}{\begin{cases} \langle c, \mathbf{tt} : e, s\rangle & \text{if } t_1{=}\mathbf{tt} \text{ and } t_2{=}\mathbf{tt} \\ \langle c, \mathbf{ff} : e, s\rangle & \text{if } t_1{=}\mathbf{ff} \text{ or } t_2{=}\mathbf{ff}, \text{ and } t_1,\ t_2{\in}\mathbf{T} \end{cases}}
\end{array}
$$

Semántica máquina abstracta 2

$$\langle \text{NEG}{:}c,\ t{:}e,\ s\rangle \ \triangleright \begin{cases} \langle c, \mathbf{ff} : e, s\rangle & \text{if } t{=}\mathbf{tt} \\ \langle c, \mathbf{tt} : e, s\rangle & \text{if } t{=}\mathbf{ff} \end{cases}$$

$$\langle \text{FETCH-}x{:}c,\ e,\ s\rangle \ \triangleright \ \langle c,\ (s\ x){:}e,\ s\rangle$$

$$\langle \text{STORE-}x{:}c,\ z{:}e,\ s\rangle \ \triangleright \ \langle c,\ e,\ s[x{\mapsto}z]\rangle \quad \text{if } z{\in}\mathbf{Z}$$

$$\langle \text{NOOP}{:}c,\ e,\ s\rangle \ \triangleright \ \langle c,\ e,\ s\rangle$$

$$\langle \text{BRANCH}(c_1,\ c_2){:}c,\ t{:}e,\ s\rangle \ \triangleright \begin{cases} \langle c_1 : c, e, s\rangle & \text{if } t{=}\mathbf{tt} \\ \langle c_2 : c, e, s\rangle & \text{if } t{=}\mathbf{ff} \end{cases}$$

$$\langle \text{LOOP}(c_1,\ c_2){:}c,\ e,\ s\rangle \ \triangleright$$

$$\langle c_1{:}\text{BRANCH}(c_2{:}\text{LOOP}(c_1,\ c_2),\ \text{NOOP}){:}c,\ e,\ s\rangle$$

Traducción de sentencias **While** 1

$$
\begin{aligned}
\mathcal{CA}[\![n]\!] &= \text{PUSH-}n \\
\mathcal{CA}[\![x]\!] &= \text{FETCH-}x \\
\mathcal{CA}[\![a_1 + a_2]\!] &= \mathcal{CA}[\![a_2]\!]{:}\mathcal{CA}[\![a_1]\!]{:}\text{ADD} \\
\mathcal{CA}[\![a_1 \star a_2]\!] &= \mathcal{CA}[\![a_2]\!]{:}\mathcal{CA}[\![a_1]\!]{:}\text{MULT} \\
\mathcal{CA}[\![a_1 - a_2]\!] &= \mathcal{CA}[\![a_2]\!]{:}\mathcal{CA}[\![a_1]\!]{:}\text{SUB} \\
\\
\mathcal{CB}[\![\texttt{true}]\!] &= \text{TRUE} \\
\mathcal{CB}[\![\texttt{false}]\!] &= \text{FALSE} \\
\mathcal{CB}[\![a_1 = a_2]\!] &= \mathcal{CA}[\![a_2]\!]{:}\mathcal{CA}[\![a_1]\!]{:}\text{EQ} \\
\mathcal{CB}[\![a_1 \leq a_2]\!] &= \mathcal{CA}[\![a_2]\!]{:}\mathcal{CA}[\![a_1]\!]{:}\text{LE} \\
\mathcal{CB}[\![\neg b]\!] &= \mathcal{CB}[\![b]\!]{:}\text{NEG} \\
\mathcal{CB}[\![b_1 \wedge b_2]\!] &= \mathcal{CB}[\![b_2]\!]{:}\mathcal{CB}[\![b_1]\!]{:}\text{AND}
\end{aligned}
$$

Traducción de sentencias **While** 2

$$
\begin{aligned}
\mathcal{CS}[\![x := a]\!] &= \mathcal{CA}[\![a]\!]\text{:STORE-}x \\
\mathcal{CS}[\![\texttt{skip}]\!] &= \text{NOOP} \\
\mathcal{CS}[\![S_1;S_2]\!] &= \mathcal{CS}[\![S_1]\!]\text{:}\mathcal{CS}[\![S_2]\!] \\
\mathcal{CS}[\![\texttt{if } b \texttt{ then } S_1 \texttt{ else } S_2]\!] &= \mathcal{CB}[\![b]\!]\text{:BRANCH}(\mathcal{CS}[\![S_1]\!],\mathcal{CS}[\![S_2]\!]) \\
\mathcal{CS}[\![\texttt{while } b \texttt{ do } S]\!] &= \text{LOOP}(\mathcal{CB}[\![b]\!],\mathcal{CS}[\![S]\!])
\end{aligned}
$$

Enlaces al código

Lenguaje While: `https://github.com/jsilvazu/While`
Máquina abstracta y traductor: `https://github.com/jsilvazu/AM`