IV - Semântica de Árvores de Derivação

4.1. Introdução

Neste capítulo estenderemos o conceito de semântica denotacional, usualmente atribuído a sentenças (programas) de uma dada linguagem para todos os morfismos de uma floresta de derivações; sendo assim, todas as (tuplas de) derivações sobre uma determinada gramática terão um significado associado a elas.

Esse estilo de semântica não acrescenta qualquer informação a respeito do comportamento do programa no sentido extensional. Acrescenta, no entanto, informações de como as construções sintáticas são usadas para se obter um comportamento desejado. São exatamente essas as informações que necessitamos para poder avaliar traduções entre linguagens no que diz respeito à manutenibilidade dos programas traduzidos.

4.2. Semântica denotacional usual

A semântica denotacional usual de uma linguagem consiste na especificação de um mapeamento que leva uma sentença desta linguagem diretamente no seu significado, chamado *denotação*. Esse mapeamento é definido por indução na estrutura dos programas, podendo ser entendido como um conjunto de funções semânticas $\{F\}_N$ indexado pelo conjunto de não-terminais da gramática tal que F_A leva o conjunto de todas as árvores de derivação de raiz A no seu respectivo significado. O domínio destas funções semânticas são *domínios semânticos*, estruturas definidas a seguir que nos permitem construir o significado dos programas da linguagem. Sugerimos [Stoy, 1977] para uma introdução completa à teoria de semântica denotacional ou [Schmidt,1986] para um introdução mais prática.

Definição 4.1. Uma categoria **Dom** de domínios semânticos é uma categoria cartesiana fechada com somas finitas e objeto número natural. Os objetos de **Dom** são

chamados de domínios semânticos. Na prática, assumimos que alguns domínios primitivos como **IN** – naturais – e **Ide** – identificadores – estejam em Obj(**Dom**).

Exemplo 4.2. O exemplo que se segue ilustra a semântica denotacional usual. A gramática escolhida é a sintaxe concreta de uma gramática de expressões infixas com ambiente manipulável através do comando **let**.

Sintaxe

PROG ::= E
E ::= E + T | T | **let** Id = E **in** E
T ::= T * F | F
F ::= (E) | N | Id

Domínios Semânticos

$\rho \in$ **Amb** = **Ide** $\rightarrow$ **IN** (ambientes)

Assinatura das funções semânticas

N: N$\rightarrow$ IN
I: Id $\rightarrow$ Ide
P: PROG $\rightarrow$ IN
E: E $\rightarrow$ Amb $\rightarrow$ IN
T: T $\rightarrow$ Amb $\rightarrow$ IN
F: F $\rightarrow$ Amb $\rightarrow$ IN

Corpo das funções semânticas

P[[E]] = E[[E]] λi.0
E[[E + T]] = E[[E]] + T[[T]]
E[[T]] = T[[T]]
E[[**let** Id = E_1 **in** E_2]] ρ = E[[E_2]] ρ [I[[Id]] $\leftarrow$E[[E_1]] ρ]
T[[T * F]] = T[[T]] * F[[F]]
T[[F]] = F[[F]]
F[[(E)]] = E[[E]]
F[[Id]] ρ= ρ(I[[Id]])
F[[N]] ρ= N[[N]]

Figura 4.1 Especificação denotacional usual.

Observamos que toda construção sintática tem um significado associado, por exemplo, E[[`10`]] = $\lambda\rho$.10 e P[[**`let A = 10 in A * A`**]] = 100, como demonstrado abaixo.

P[[**let** A = 10 **in** A * A]] =
E[[**let** A = 10 **in** A * A]] λi.0 =
E[[A * A]] λi.0 [A$\leftarrow$ E[[10]] λi.0] =

$$
\begin{gathered}
E[\![\ A * A]\!]\ \lambda i.0\ [A \leftarrow N[\![10]\!]\ ] = \\
E[\![\ A * A]\!]\ \lambda i.0\ [A \leftarrow 10\ ] = \\
T[\![\ A * A]\!]\ \lambda i.0\ [A \leftarrow 10\ ] = \\
T[\![\ A\ ]\!]\ \lambda i.0\ [I[\![A]\!] \leftarrow 10\ ] * F[\![\ A\ ]\!]\ \lambda i.0\ [I[\![A]\!] \leftarrow 10\ ] = \\
F[\![\ A\ ]\!]\ \lambda i.0\ [I[\![A]\!] \leftarrow 10\ ] * F[\![\ A\ ]\!]\ \lambda i.0\ [I[\![A]\!] \leftarrow 10\ ] = \\
F[\![\ A\ ]\!]\ \lambda i.0\ [I[\![A]\!] \leftarrow 10\ ] * F[\![\ A\ ]\!]\ \lambda i.0\ [I[\![A]\!] \leftarrow 10\ ] = \\
\lambda i.0\ [I[\![A]\!] \leftarrow 10\ ]\ (I[\![A]\!]) * \lambda i.0\ [I[\![A]\!] \leftarrow 10\ ]\ (I[\![A]\!]) = \\
10 * 10 = \\
100
\end{gathered}
$$

Figura 4.2 Avaliação da semântica de '`let A = 10 in A * A`'.

4.3. Uma interpretação concreta de uma floresta de derivações

Seja $(\mathbf{F}, \Sigma, \phi, (C,\square,\mathbf{1})) = L_{\square}(G,\Sigma,\phi)$. Definimos $I:\mathbf{C}\rightarrow\mathbf{Sets}$ da seguinte forma:

- $I(\mathbf{1}) = \{*\}$. I leva o elemento neutro de $\square$ no unitário em **Sets**.
- $I\ (N \in Obj(C))$ = Conjunto de todas as árvores de derivação em G cuja raiz é N e as folhas são terminais.
- $I\ (r_i:X_1\square\ldots\square X_n\rightarrow Y \in Arr(C))$ = Função f que dadas n árvores $A_1,\ldots,A_n$, de raízes respectivamente $X_1\ldots X_n$, retorna a árvore A de raiz Y, de acordo com a figura 4.3.

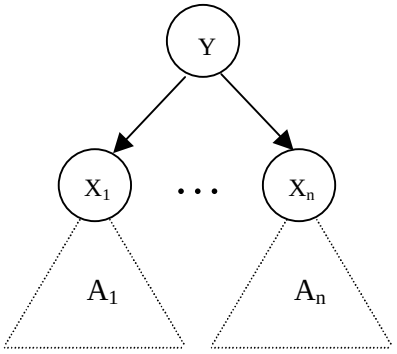

Figura 4.3 Resultado de $f(A_1,\ldots,A_n)$.

Além disso, como a imagem de I é **Sets**, cada árvore do conjunto de árvores I(N) corresponde a um morfismo $a:\{*\}\rightarrow I(N)$, isto é, a é um *elemento* de I(N), no sentido categórico. Esse morfismo é a composição das aplicações de funções necessárias à construção da árvore a partir do símbolo N.

Exemplo 4.3. A figura 4.4 ilustra a construção de um elemento na interpretação concreta I(**C**) para a gramática G_{EXP} definida no exemplo 3.19. Nesta figura $m_1(*)$ = "ID⇒id", m_2("ID⇒id") = "E⇒ID⇒id", m(*) = "E⇒ID⇒id"

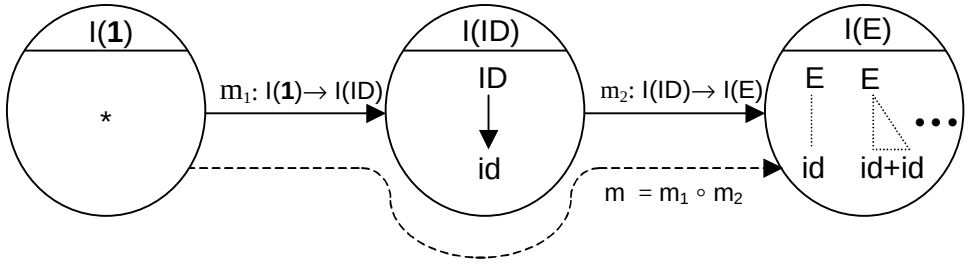

Figura 4.4 Exemplo de I(C).

Através das assinaturas das funções semânticas em uma especificação denotacional usual, sabemos em qual domínio semântico serão mapeados os significados das árvores de derivação completas de raiz N. Podemos entender esse mapeamento como a especificação da componente objeto de um funtor $[\![\_]\!]_I$:I(**C**)→**Dom**. A componente morfismo desse funtor deve ser tal que mantenha o significado descrito pela especificação denotacional usual. A grande vantagem será que esse funtor nos permitirá designar significado para (tuplas de) árvores de derivação parciais, isto é, a um morfismo qualquer de I(**C**).

4.4. Semântica de árvores de derivação

Definição 4.4. Uma **semântica de árvores de derivação** (ou simplesmente semântica de árvores) para (F, Σ, ϕ, **C**) = $L_{\square}$(G,Σ,ϕ) é um funtor $[\![\_]\!]$:**C**→**Dom**. Pode ser mais fácil entender $[\![\_]\!]$ como I ∘ $[\![\_]\!]_I$. Onde I:**C**→**Sets** é a interpretação concreta definida anteriormente e $[\![\_]\!]_I$ é um funtor $[\![\_]\!]_I$:I(**C**)→**Dom**.

Exemplo 4.5. A figura 4.5 ilustra a semântica de árvores de derivação. A linguagem escolhida é a mesma do exemplo 4.2. Cada regra da gramática é mapeada em uma função que constrói o significado do seu lado esquerdo em função do significado de

seu lado direito (sub-árvores). Todas as demais arestas são obtidas através de extensão funtorial, preservando $\Box$ e $\mathbf{1}$.

Sintaxe

PROG ::= E
E ::= E + T | T | **let** Id = E **in** E
T ::= T * F | F
F ::= (E) | N | Id

Domínios Semânticos

$\rho \in \mathbf{Amb} = \mathbf{Ide} \to \mathbb{N}$ (ambientes)

Componente objeto de [[_]]

$[\![\text{PROG}]\!] = \mathbb{N}$
$[\![\text{N}]\!] = \mathbb{N}$
$[\![\text{Id}]\!] = \text{Ide}$
$[\![\text{E}]\!] = \text{Amb} \to \mathbb{N}$
$[\![\text{T}]\!] = \text{Amb} \to \mathbb{N}$
$[\![\text{F}]\!] = \text{Amb} \to \mathbb{N}$

Componente morfismo de [[_]]

$[\![\text{PROG} \to \text{E}]\!]\ f = f\ \lambda i.0$
$[\![\text{E} \to \text{E} + \text{T}]\!]\ f\ g = f + g$
$[\![\text{E} \to \text{T}]\!]\ f = f$
$[\![\text{E} \to \textbf{let}\ \text{Id} = \text{E}_1\ \textbf{in}\ \text{E}_2]\!]\ \text{Id}\ f\ g =$
$\lambda\rho.\ g\ (\rho\ [\ [\![\text{Id}]\!] \leftarrow f(\rho)\ ]\ )$
$[\![\text{T} \to \text{T} * \text{F}]\!]\ f\ g = f * g$
$[\![\text{T} \to \text{F}]\!]\ f = f$
$[\![\text{F} \to (\text{E})]\!]\ f = f$
$[\![\text{F} \to \text{N}]\!]\ n = [\![n]\!]$
$[\![\text{F} \to \text{Id}]\!]\ \text{Id} = \lambda\rho.\rho([\![\text{Id}]\!])$

} ver explicação abaixo

Figura 4.5 Exemplo de semântica de árvores de derivação.

Observe que $[\![\text{PROG} \to \text{E}]\!]$ é um mapeamento $[\![\text{E}]\!]$, de assinatura $\text{Amb} \to \mathbb{N}$ em $[\![\text{PROG}]\!]$, de assinatura $\mathbb{N}$. Seu resultado é aplicar a semântica da sub-árvore de raiz E ao ambiente inicial, $\lambda i.0$, para obter a semântica da árvore de raiz PROG, daí a equação $[\![\text{PROG} \to \text{E}]\!]\ f = f\ \lambda i.0$. Da mesma forma, $[\![\text{E} \to \text{E} + \text{T}]\!]$ calcula a semântica da árvore de raiz E em função da semântica de suas sub-árvores de raízes E e T. Daí a equação $[\![\text{E} \to \text{E} + \text{T}]\!]\ f\ g = f + g$, ou então da forma mais explícita: $[\![\text{E} \to \text{E} + \text{T}]\!] = \lambda f.\ \lambda g.\lambda\rho.f(\rho)+g(\rho)$.

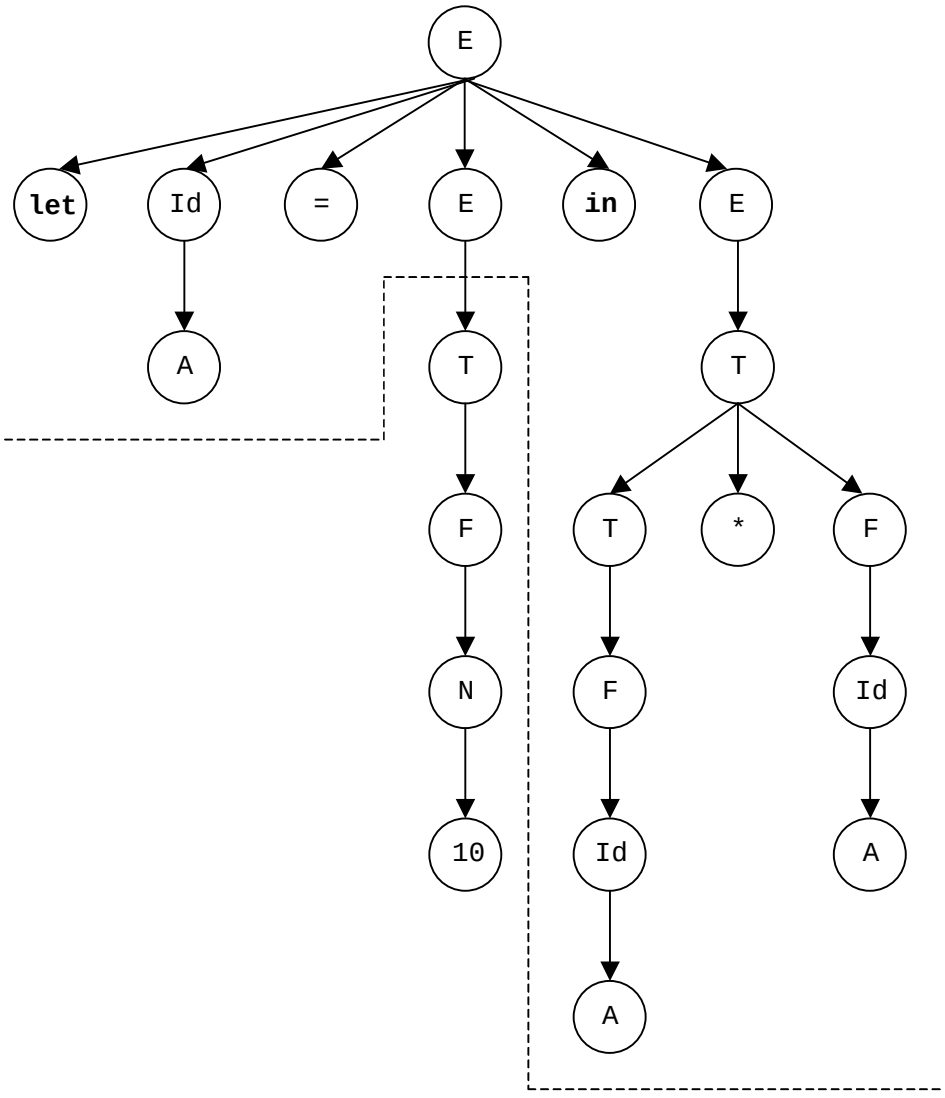

Figura 4.6 Árvore sintática de "**let** A = 10 **in** A * A".

Considere agora a expressão: **let** A = 10 **in** A * A. A figura 4.6 exibe sua árvore de derivação. Sua semântica é $[\![$E ⇒ **let** A = 10 **in** A * A$]\!]$: $1 \to \mathbb{N}$, cujo valor é $\lambda\rho.100$. No entanto, o código dessa expressão exprime muito mais do que seu valor. Por exemplo, o código nos mostra que o valor da expressão é obtido elevando-se o valor de A ao quadrado. Usando-se a semântica de árvores de derivação, podemos calcular $[\![$E ⇒ **let** A = E **in** A * A$]\!]$: $(Amb \to \mathbb{N}) \to \mathbb{N}$ (que corresponde ao significado da árvore de derivação acima da linha pontilhada na figura 4.6), cujo valor é: $\lambda f.\lambda\rho.(f(\rho))^2$.

Caso a expressão original fosse traduzida para “100”, certamente teria sua “semântica” preservada no sentido usual, i.e., extensional. No entanto, essa tradução não preservaria a semântica de suas árvores de derivação, pois não há nenhuma estrutura em “100” com significado $\lambda f.\lambda \rho.(f(\rho))^2$.

No próximo capítulo, argumentaremos que a existência de estruturas com semântica equivalente no programa traduzido é uma condição necessária para a preservação da manutenibilidade.