

Semântica Denotacional

Maria João Frade

HASLab - INESC TEC
Departamento de Informática, Universidade do Minho

2021/2022

Semântica Denotacional

- O significado de um programa é descrito num domínio matemático abstracto.
- O foco desta abordagem é *no efeito* da execução de um programa e não na forma como essa execução é feita.
- A ideia é definir uma *função semântica* para cada categoria sintáctica.
 - ▶ A função semântica mapeia cada construção sintáctica num objecto matemático que descreve o efeito de executar essa construção.
- A característica essencial da semântica denotacional é o seu carácter *composicional*.

Composicionalidade

Na abordagem denotacional as funções semânticas devem ser definidas de forma *composicional*:

- uma cláusula para cada elemento de base da categoria sintáctica
- a semântica de cada elemento composto é definida à custa da semântica dos seus constituintes imediatos

A semântica das expressões aritméticas e booleanas da linguagem **While** foi definida denotacionalmente:

$$\mathcal{A} : \mathbf{Aexp} \rightarrow (\mathbf{State} \rightarrow \mathbf{Z}) \quad \mathcal{B} : \mathbf{Bexp} \rightarrow (\mathbf{State} \rightarrow \mathbf{T})$$

Mas as funções semânticas induzidas pela semântica operacional (big-step e small-step) para os comandos *não são definidas composicionalmente*.

$$\mathcal{S}_{\text{ns}} : \mathbf{Stm} \rightarrow (\mathbf{State} \hookrightarrow \mathbf{State}) \quad \mathcal{S}_{\text{sos}} : \mathbf{Stm} \rightarrow (\mathbf{State} \hookrightarrow \mathbf{State})$$

Semântica denotacional para a linguagem **While**

O efeito de executar um comando é alterar o estado, mas a execução de um comando pode não terminar. O significado de um comando será dado por uma *função parcial sobre os estados*.

$$\mathcal{S}_{\text{ds}} : \mathbf{Stm} \rightarrow (\mathbf{State} \rightharpoonup \mathbf{State})$$

- Para a atribuição

$$\mathcal{S}_{\text{ds}}[x := a] s = s[x \mapsto \mathcal{A}[a] s]$$

- Para o comando skip

$$\mathcal{S}_{\text{ds}}[\text{skip}] = \text{id}$$

id é a função identidade.

$$\mathcal{S}_{\text{ds}}[\text{skip}] s = s$$

Semântica denotacional para a linguagem **While**

- Para a sequenciação de comandos

$$\mathcal{S}_{ds}[[S_1 ; S_2]] = \mathcal{S}_{ds}[[S_2]] \circ \mathcal{S}_{ds}[[S_1]]$$

$$\begin{aligned} \mathcal{S}_{ds}[[S_1 ; S_2]]s &= (\mathcal{S}_{ds}[[S_2]] \circ \mathcal{S}_{ds}[[S_1]])s \\ &= \begin{cases} s'' & \text{if there exists } s' \text{ such that } \mathcal{S}_{ds}[[S_1]]s = s' \\ & \text{and } \mathcal{S}_{ds}[[S_2]]s' = s'' \\ \underline{\text{undef}} & \text{if } \mathcal{S}_{ds}[[S_1]]s = \underline{\text{undef}} \\ & \text{or if there exists } s' \text{ such that } \mathcal{S}_{ds}[[S_1]]s = s' \\ & \text{but } \mathcal{S}_{ds}[[S_2]]s' = \underline{\text{undef}} \end{cases} \end{aligned}$$

Semântica denotacional para a linguagem **While**

- Para comandos condicionais

$$\mathcal{S}_{ds}[[\text{if } b \text{ then } S_1 \text{ else } S_2]] = \text{cond}(\mathcal{B}[[b]], \mathcal{S}_{ds}[[S_1]], \mathcal{S}_{ds}[[S_2]])$$

onde **cond** é a função auxiliar

$$\text{cond} : (\text{State} \rightarrow \mathbf{T}) \times (\text{State} \hookrightarrow \text{State}) \times (\text{State} \hookrightarrow \text{State}) \rightarrow (\text{State} \hookrightarrow \text{State})$$

$$\text{cond}(p, g_1, g_2) s = \begin{cases} g_1 s & \text{se } p s = \mathbf{tt} \\ g_2 s & \text{se } p s = \mathbf{ff} \end{cases}$$

Semântica denotacional para a linguagem **While**

$$\begin{aligned} \mathcal{S}_{ds}[[\text{if } b \text{ then } S_1 \text{ else } S_2]]s &= \text{cond}(\mathcal{B}[[b]], \mathcal{S}_{ds}[[S_1]], \mathcal{S}_{ds}[[S_2]])s \\ &= \begin{cases} s' & \text{if } \mathcal{B}[[b]]s = \mathbf{tt} \text{ and } \mathcal{S}_{ds}[[S_1]]s = s' \\ & \text{or if } \mathcal{B}[[b]]s = \mathbf{ff} \text{ and } \mathcal{S}_{ds}[[S_2]]s = s' \\ \underline{\text{undef}} & \text{if } \mathcal{B}[[b]]s = \mathbf{tt} \text{ and } \mathcal{S}_{ds}[[S_1]]s = \underline{\text{undef}} \\ & \text{or if } \mathcal{B}[[b]]s = \mathbf{ff} \text{ and } \mathcal{S}_{ds}[[S_2]]s = \underline{\text{undef}} \end{cases} \end{aligned}$$

Semântica denotacional para a linguagem **While**

- O efeito de executar **while** b do S deve ser igual ao de executar

$$\text{if } b \text{ then } \{S; \text{while } b \text{ do } S\} \text{ else skip}$$

Por isso, uma primeira proposta poderá ser

$$\mathcal{S}_{ds}[[\text{while } b \text{ do } S]] = \text{cond}(\mathcal{B}[[b]], \mathcal{S}_{ds}[[\text{while } b \text{ do } S]] \circ \mathcal{S}_{ds}[[S]], \text{id})$$

Mas esta definição **não é composicional!**

- Contudo, esta equação diz-nos que $\mathcal{S}_{ds}[[\text{while } b \text{ do } S]]$ deve ser um **ponto fixo** da funcional F definida por

$$Fg = \text{cond}(\mathcal{B}[[b]], g \circ \mathcal{S}_{ds}[[S]], \text{id})$$

isto é, $\mathcal{S}_{ds}[[\text{while } b \text{ do } S]] = F(\mathcal{S}_{ds}[[\text{while } b \text{ do } S]])$. Deste modo teremos uma definição composicional de \mathcal{S}_{ds} .

Semântica denotacional para a linguagem **While**

- Para os ciclos deverá então ser

$$\mathcal{S}_{ds}[\text{while } b \text{ do } S] = \text{FIX } F$$

$$\text{onde } Fg = \text{cond}(\mathcal{B}[b], g \circ \mathcal{S}_{ds}[S], \text{id})$$

A função auxiliar **FIX** tem tipo

$$\text{FIX} : ((\text{State} \hookrightarrow \text{State}) \rightarrow (\text{State} \hookrightarrow \text{State})) \rightarrow (\text{State} \hookrightarrow \text{State})$$

Semântica denotacional para a linguagem **While**

Mas a definição de $\mathcal{S}_{ds}[\text{while } b \text{ do } S]$ tem ainda que lidar com os seguintes **problemas**:

- Há funcionais que têm *mais do que um ponto fixo*. Por exemplo,

$$(F'g)s = \begin{cases} g\ s & \text{se } sx \neq 0 \\ s & \text{se } sx = 0 \end{cases}$$

Qualquer função $g : \text{State} \hookrightarrow \text{State}$ tal que $gs = s$ se $sx = 0$ será um ponto fixo de F' . Exemplos:

$$g's = \begin{cases} s & \text{se } sx = 0 \\ \text{undef} & \text{se } sx \neq 0 \end{cases} \quad g''s = \begin{cases} s & \text{se } sx = 0 \\ s[x \mapsto 0] & \text{se } sx \neq 0 \end{cases}$$

Temos $(F'g')s = \dots = g's$ e $(F'g'')s = \dots = g''s$.

Ou seja, *g' e g'' são dois pontos fixos de F'* , pois $F'g' = g'$ e $F'g'' = g''$.

Semântica denotacional para a linguagem **While**

- Há funcionais que *não têm pontos fixos*. Por exemplo, caso $g_1 \neq g_2$,

$$F''g = \begin{cases} g_1 & \text{se } g = g_2 \\ g_2 & \text{se } g \neq g_2 \end{cases}$$

Se existisse uma função g_0 ponto fixo de F'' , teríamos $F''g_0 = g_0$ e, nesse caso,

$$F''g_0 = \begin{cases} g_1 & \text{se } g_0 = g_2 \\ g_2 & \text{se } g_0 \neq g_2 \end{cases} = g_0$$

teríamos $g_1 = g_2$, o que contradiz a nossa hipótese.

Logo, *a funcional F'' não tem pontos fixos*.

Semântica denotacional para a linguagem **While**

Solução:

- Impor condições aos pontos fixos e mostrar que há no máximo um ponto fixo que satisfaz essas condições.
- Provar que todas as funcionais que são geradas pelos comandos “while” têm um ponto fixo que satisfaz essas condições.

Conjuntos parcialmente ordenados

Para definir os requisitos que garantem a existência do ponto fixo desejado $\text{FIX } F$ teremos que introduzir alguns conceitos e resultados.

Um *conjunto parcialmente ordenado (poset)* é um par (D, \sqsubseteq) , sendo D um conjunto e \sqsubseteq uma relação binária em D que é

- reflexiva: $\forall d \in D, d \sqsubseteq d$
- transitiva: $\forall d_1, d_2, d_3 \in D$, se $d_1 \sqsubseteq d_2$ e $d_2 \sqsubseteq d_3$ então $d_1 \sqsubseteq d_3$
- anti-simétrica: $\forall d_1, d_2 \in D$, se $d_1 \sqsubseteq d_2$ e $d_2 \sqsubseteq d_1$ então $d_1 = d_2$

A relação \sqsubseteq diz-se uma *ordem parcial*.

- $d \in D$ diz-se *o menor elemento* (ou *mínimo*) de D , se para todo $d' \in D, d \sqsubseteq d'$.
- Se o menor elemento existir, então ele é *único*.

Conjuntos parcialmente ordenados

Lema

Seja \sqsubseteq a seguinte relação binária em $\text{State} \leftrightarrow \text{State}$:

$$g_1 \sqsubseteq g_2 \quad \text{sse} \quad \text{para todo } s, s', \text{ se } g_1 s = s' \text{ então } g_2 s = s'$$

- $(\text{State} \leftrightarrow \text{State}, \sqsubseteq)$ é um *conjunto parcialmente ordenado*.
- A função $\perp: \text{State} \leftrightarrow \text{State}$ definida por

$$\perp s = \text{undef}, \text{ para todo } s$$

é o *menor elemento* de $\text{State} \leftrightarrow \text{State}$.

Pontos fixos

O que vamos exigir de $\text{FIX } F$ é que:

- $\text{FIX } F$ seja um *ponto fixo* de F , isto é, $F(\text{FIX } F) = \text{FIX } F$
- $\text{FIX } F$ seja o *menor ponto fixo* de F , isto é,

$$\text{se } F g = g \text{ então } \text{FIX } F \sqsubseteq g$$

Vamos agora ver como se garante que para todas as funcionais F , existe o menor ponto fixo.

Conjuntos parcialmente ordenados

Seja (D, \sqsubseteq) um *conjunto parcialmente ordenado*, e $Y \subseteq D$

- $d \in D$ é um *limite superior* de Y se $\forall d' \in Y. d' \sqsubseteq d$
- $d \in D$ é um *limite superior mínimo* de Y se é um limite superior de Y e para todo o d' que seja um limite superior de $Y, d \sqsubseteq d'$.
- O limite superior mínimo de Y se existir é único e denota-se $\bigsqcup Y$.
- $Y \neq \emptyset$ é uma *cadeia* se $\forall d_1, d_2 \in Y. d_1 \sqsubseteq d_2$ ou $d_2 \sqsubseteq d_1$
- (D, \sqsubseteq) diz-se uma *ordem parcial completa (CPO)* se existe $\bigsqcup Y$ para todas as cadeias Y . Também chamado de *pré-domínio*.
- Caso um CPO tenha um elemento *mínimo* (\perp) diz-se que é um *BCPO (bottom complete partial order)*. Também chamado de *domínio*.

Conjuntos parcialmente ordenados

Lema

$(\text{State} \hookrightarrow \text{State}, \sqsubseteq)$ é um domínio.

O limite superior mínimo de uma cadeia Y é tal que

$$(\bigsqcup Y) s = s' \text{ sse } g s = s' \text{ para algum } g \in Y$$

Funções monótonas

Sejam (D, \sqsubseteq) e (D', \sqsubseteq') domínios. A função $f : D \rightarrow D'$ diz-se *monótona* sse

$$\forall d_1, d_2 \in D. \quad d_1 \sqsubseteq d_2 \Rightarrow f d_1 \sqsubseteq' f d_2$$

Lema

Sejam (D, \sqsubseteq) , (D', \sqsubseteq') e (D'', \sqsubseteq'') domínios e $f : D \rightarrow D'$ e $f' : D' \rightarrow D''$ funções monótonas. Então $f' \circ f : D \rightarrow D''$ é uma função monótona.

Lema

Sejam (D, \sqsubseteq) e (D', \sqsubseteq') domínios e $f : D \rightarrow D'$ uma função monótona.

- Se Y é uma cadeia de D então $f Y$ é uma cadeia de D' .
- $\bigsqcup (f Y) \sqsubseteq' f (\bigsqcup Y)$

Funções contínuas

As funções monótonas podem não preservar limites superiores mínimos.

Sejam (D, \sqsubseteq) e (D', \sqsubseteq') domínios. A função $f : D \rightarrow D'$ diz-se *contínua* se é monótona e para todas as cadeias Y ,

$$\bigsqcup (f Y) = f (\bigsqcup Y)$$

Se se verificar que $\perp = f \perp$, então f diz-se *estrita*.

Lema

Sejam (D, \sqsubseteq) , (D', \sqsubseteq') e (D'', \sqsubseteq'') domínios e $f : D \rightarrow D'$ e $f' : D' \rightarrow D''$ funções contínuas. Então $f' \circ f : D \rightarrow D''$ é uma função contínua.

O menor ponto fixo

Teorema

Seja $f : D \rightarrow D$ uma função contínua sobre o domínio (D, \sqsubseteq) , com elemento mínimo \perp . Então

$$\text{FIX } f = \bigsqcup \{ f^n \perp \mid n \geq 0 \}$$

define um elemento de D e este elemento é o *menor ponto fixo* de f .

Estamos a usar a seguinte notação

$$\begin{aligned} f^0 &= \text{id} \\ f^{n+1} &= f \circ f^n, \text{ para } n \geq 0 \end{aligned}$$

Semântica denotacional para a linguagem **While**

Estamos agora em condições de garantir que a cláusula proposta para a interpretação denotacional dos ciclos é uma boa definição.

$$\begin{aligned} \mathcal{S}_{ds}[\text{while } b \text{ do } S] &= \text{FIX } F \\ \text{onde } F g &= \text{cond}(\mathcal{B}[b], g \circ \mathcal{S}_{ds}[S], \text{id}) \end{aligned}$$

Precisamos apenas de mostrar que a funcional F é contínua.

Note que $F g = F_1 (F_2 g)$

$$\begin{aligned} \text{com } F_1 g &= \text{cond}(\mathcal{B}[b], g, \text{id}) \\ F_2 g &= g \circ \mathcal{S}_{ds}[S]. \end{aligned}$$

Semântica denotacional para a linguagem **While**

Lema

Seja $g_0 : \text{State} \hookrightarrow \text{State}$, $p : \text{State} \hookrightarrow \mathbf{T}$ e $F g = \text{cond}(p, g, g_0)$. Então F é contínua.

Lema

Seja $g_0 : \text{State} \hookrightarrow \text{State}$ e $F g = g \circ g_0$. Então F é contínua.

Com estes dois lemas estamos em condições de garantir que

$$F g = F_1 (F_2 g) = \text{cond}(\mathcal{B}[b], g \circ \mathcal{S}_{ds}[S], \text{id})$$

é uma função contínua, pois é a composição de duas funções contínuas.

Semântica denotacional para a linguagem **While**

Proposição

$$\mathcal{S}_{ds} : \text{Stm} \rightarrow (\text{State} \hookrightarrow \text{State})$$

$$\mathcal{S}_{ds}[x := a] s = s[x \mapsto \mathcal{A}[a] s]$$

$$\mathcal{S}_{ds}[\text{skip}] = \text{id}$$

$$\mathcal{S}_{ds}[S_1 ; S_2] = \mathcal{S}_{ds}[S_2] \circ \mathcal{S}_{ds}[S_1]$$

$$\mathcal{S}_{ds}[\text{if } b \text{ then } S_1 \text{ else } S_2] = \text{cond}(\mathcal{B}[b], \mathcal{S}_{ds}[S_1], \mathcal{S}_{ds}[S_2])$$

$$\mathcal{S}_{ds}[\text{while } b \text{ do } S] = \text{FIX } F$$

$$\text{onde } F g = \text{cond}(\mathcal{B}[b], g \circ \mathcal{S}_{ds}[S], \text{id})$$

define uma função total.

Prova: Por indução na estrutura dos comandos.

Exercício

Exercício

Considere a semântica denotacional do seguinte programa

$$y := 1; \text{ while } x \neq 1 \text{ do } \{y := y * x; x := x - 1\}$$

Aplique a função resultante a um estado s_0 tal que $s_0 x = 3$, e indique o valor da variável y no estado de chegada.

Sugestões:

- 1 Construa $\mathcal{S}_{ds}[y := 1; \text{ while } x \neq 1 \text{ do } \{y := y * x; x := x - 1\}]$ identificando a funcional F envolvida.
- 2 Calcule as várias funções $F^n \perp$ usadas na definição de $\text{FIX } F$ e apresente uma definição de $\text{FIX } F$.
- 3 Tem agora todos os dados para calcular o valor de y no estado de chegada.

Equivalência semântica de programas

Podemos introduzir uma noção de equivalência semântica de programas baseada na semântica denotacional.

Sejam $S_1, S_2 \in \mathbf{Stm}$.

S_1 e S_2 são *semanticamente equivalentes* sse $\mathcal{S}_{\text{ds}}[[S_1]] = \mathcal{S}_{\text{ds}}[[S_2]]$

Exercício

Mostre que os seguintes programas são semanticamente equivalentes:

- $S; \text{skip}$ e S
- $S_1; \{S_2; S_3\}$ e $\{S_1; S_2\}; S_3$
- $\text{while } b \text{ do } S$ e $\text{if } b \text{ then } \{S; \text{while } b \text{ do } S\} \text{ else skip}$

Equivalência entre semânticas

Como é que a semântica denotacional se relaciona com a operacional?

Teorema

Para todo $S \in \mathbf{Stm}$, $\mathcal{S}_{\text{sos}}[[S]] = \mathcal{S}_{\text{ds}}[[S]]$.

Prova: A prova baseia-se nos seguintes lemas

Para todo $S \in \mathbf{Stm}$, $\mathcal{S}_{\text{sos}}[[S]] \subseteq \mathcal{S}_{\text{ds}}[[S]]$.

Prova: Tem que se mostrar que se $\langle S, s \rangle \Rightarrow^* s'$ então $\mathcal{S}_{\text{ds}}[[S]] s = s'$.

Para todo $S \in \mathbf{Stm}$, $\mathcal{S}_{\text{ds}}[[S]] \subseteq \mathcal{S}_{\text{sos}}[[S]]$.

Prova: Por indução estrutural em S .

Qual será a relação entre \mathcal{S}_{ds} e \mathcal{S}_{ns} ?