Semântica Denotacional

Maria João Frade

HASLab - INESC TEC
Departamento de Informática, Universidade do Minho

2019/2020

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 caracteristica 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:

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

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

$$\mathcal{S}_{\mathrm{ns}} : \mathbf{Stm} \to (\mathbf{State} \hookrightarrow \mathbf{State}) \qquad \mathcal{S}_{\mathrm{sos}} : \mathbf{Stm} \to (\mathbf{State} \hookrightarrow \mathbf{State})$$

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.

$$\mathcal{S}_{\mathrm{ds}} : \mathbf{Stm} \to (\mathbf{State} \hookrightarrow \mathbf{State})$$

- Para a atribuição

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

- Para o comando `skip`

$$\mathcal{S}_{\mathrm{ds}}[\![\texttt{skip}]\!] = \mathrm{id}$$

id é a função identidade.

$$\mathcal{S}_{\mathrm{ds}}[\![\texttt{skip}]\!]\ s = s$$

Semântica denotacional para a linguagem **While**

- Para a sequenciação de comandos

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

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

Semântica denotacional para a linguagem **While**

- Para comandos condicionais

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

onde cond é a função auxiliar

$$\begin{aligned} \mathrm{cond}: \ & (\mathbf{State} \to \mathbf{T}) \times (\mathbf{State} \hookrightarrow \mathbf{State}) \times (\mathbf{State} \hookrightarrow \mathbf{State}) \\ & \to (\mathbf{State} \hookrightarrow \mathbf{State}) \end{aligned}$$

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

Semântica denotacional para a linguagem **While**

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

Semântica denotacional para a linguagem **While**

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

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

  Por isso, uma primeira proposta poderá ser

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

  Mas esta definição não é composicional!

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

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

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

Semântica denotacional para a linguagem **While**

- Para os ciclos deverá então ser

$$\mathcal{S}_{\rm ds}[\![\texttt{while}\ b\ \texttt{do}\ S]\!] = \mathrm{FIX}\ F$$

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

A função auxiliar FIX tem tipo

$$\mathrm{FIX} : ((\mathbf{State} \hookrightarrow \mathbf{State}) \to (\mathbf{State} \hookrightarrow \mathbf{State})) \to (\mathbf{State} \hookrightarrow \mathbf{State})$$

Semântica denotacional para a linguagem **While**

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

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

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

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

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

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

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**

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

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

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

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

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

teriamos $g_1 = g_2$, o que contradiz a nossa hipótese.
Logo, a funcional F'' não tem pontos fixos.

Semântica denotacional para a linguagem **While**

Solução:

- Impôr condições aos pontos fixos e mostrar que há no máximo um ponto fixo que safisaz 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 requistos que garantem a existência do ponto fixo desejado $\mathrm{FIX}\ F$ 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 $\mathbf{State} \hookrightarrow \mathbf{State}$:

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

- $(\mathbf{State} \hookrightarrow \mathbf{State}, \sqsubseteq)$ é um conjunto parcialmente ordenado.
- A função $\bot$: $\mathbf{State} \hookrightarrow \mathbf{State}$ definida por

$$\bot\ \ s = \underline{\mathrm{undef}} \quad , \text{para todo } s$$

é o menor elemento de $\mathbf{State} \hookrightarrow \mathbf{State}$.

Pontos fixos

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

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

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

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

Conjuntos parcialmente ordenados

Seja $(D, \sqsubseteq)$ um <u>conjunto parcialmente ordenado</u>, 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* $(\bot)$ diz-se que é um *BCPO (bottom complete partial order)*. Também chamado de *domínio*.

Conjuntos parcialmente ordenados

Lema

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

O limite superior mínimo de uma cadeia Y é tal que

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

Funções contínuas

As funções monótonas podem não preservar limites superiores mínimos.

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

Funções monótonas

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

Lema

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

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

O menor ponto fixo

Teorema

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

$$\mathrm{FIX}\ f = \bigsqcup\{\, f^n \perp \mid\; n \geq 0 \,\}$$

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

Estamos a usar a seguinte notação

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

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.

$$\mathcal{S}_{\mathrm{ds}}[\![\texttt{while}\ b\ \texttt{do}\ S]\!] = \mathrm{FIX}\ F \quad \text{onde} \quad F\ g = \mathrm{cond}(\mathcal{B}[\![b]\!]\,, g \circ \mathcal{S}_{\mathrm{ds}}[\![S]\!]\,, \mathrm{id})$$

Precisamos apenas de mostrar que a funcional F é contínua.

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

$$\text{com} \quad F_1\, g = \mathrm{cond}(\mathcal{B}[\![b]\!]\,, g, \mathrm{id}) \qquad F_2\ g = g \circ \mathcal{S}_{\mathrm{ds}}[\![S]\!].$$

Semântica denotacional para a linguagem **While**

Lema

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

Lema

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

Com estes dois lemas estamos em condições de garantir que

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

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

Semântica denotacional para a linguagem **While**

Proposição

$$\mathcal{S}_{\mathrm{ds}} : \mathbf{Stm} \to (\mathbf{State} \hookrightarrow \mathbf{State})$$

$$\begin{aligned}
&\mathcal{S}_{\mathrm{ds}}[\![x := a]\!]\ s = s[x \mapsto \mathcal{A}[\![a]\!]\ s] \\
&\mathcal{S}_{\mathrm{ds}}[\![\texttt{skip}]\!] = \mathrm{id} \\
&\mathcal{S}_{\mathrm{ds}}[\![S_1\,;\,S_2]\!] = \mathcal{S}_{\mathrm{ds}}[\![S_2]\!] \circ \mathcal{S}_{\mathrm{ds}}[\![S_1]\!] \\
&\mathcal{S}_{\mathrm{ds}}[\![\texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2]\!] = \mathrm{cond}(\mathcal{B}[\![b]\!]\,, \mathcal{S}_{\mathrm{ds}}[\![S_1]\!]\,, \mathcal{S}_{\mathrm{ds}}[\![S_2]\!]) \\
&\mathcal{S}_{\mathrm{ds}}[\![\texttt{while}\ b\ \texttt{do}\ S]\!] = \mathrm{FIX}\ F \\
&\qquad \text{onde} \quad F\ g = \mathrm{cond}(\mathcal{B}[\![b]\!]\,, g \circ \mathcal{S}_{\mathrm{ds}}[\![S]\!]\,, \mathrm{id})
\end{aligned}$$

define uma função total.

Prova: Por indução na estrutura dos comandos.

Exercício

Exercício

Considere a semântica denotacional do seguinte programa

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

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 $\mathcal{S}_{\mathrm{ds}}[\![y := 1\,;\ \texttt{while}\ x \neq 1\ \texttt{do}\ \{y := y * x\,;\ x := x - 1\}]\!]$ identificando a funcional F envolvida.
2. Calcule as várias funções $F^n\ \bot$ usadas na definição de $\mathrm{FIX}\,F$ e apresente uma definição de $\mathrm{FIX}\,F$.
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 $\mathcal{S}_{\mathrm{ds}}[\![S_1]\!] = \mathcal{S}_{\mathrm{ds}}[\![S_2]\!]$

Exercício

Mostre que os seguintes programas são semânticamente equivalentes:

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

Equivalência entre semânticas

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

Teorema

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

Prova: A prova baseia-se nos seguintes lemas

Para todo $S \in \mathbf{Stm}$, $\mathcal{S}_{\mathrm{sos}}[\![S]\!] \sqsubseteq \mathcal{S}_{\mathrm{ds}}[\![S]\!]$.

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

Para todo $S \in \mathbf{Stm}$, $\mathcal{S}_{\mathrm{ds}}[\![S]\!] \sqsubseteq \mathcal{S}_{\mathrm{sos}}[\![S]\!]$.

Prova: Por indução estrutural em S.

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