Sumários (2007-08): Semântica da Programação

Luís Soares Barbosa

Programa.

- Introdução. Sintaxe e semântica. Semântica operacional, denotacional e axiomática.
- Esquemas de definição e prova por indução.
- Introdução à semântica operacional de uma linguagem imperativa.
- Introdução à semântica denotacional e teoria dos domínios.
- Equivalência entre as semânticas operacional e denotacional de uma linguagem imperativa.
- Semântica operacional e denotacional de uma linguagem funcional.

Bibliografia.

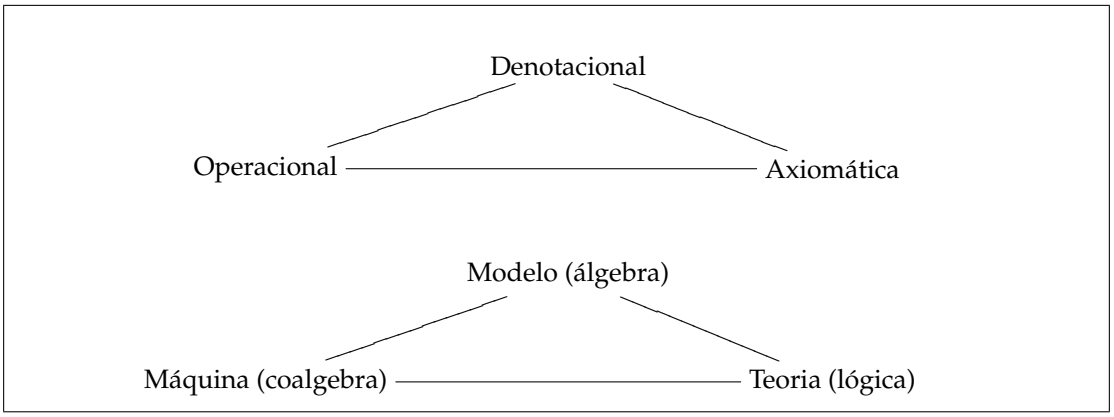
- G. Winskel. *The formal semantics of programming languages*. MIT Press. 1993.
- M. Hennessy. *The semantics of programming languages*. Wiley. 1990.
- B. Pierce. *Types and programming languages*. MIT Press. 2000.
- T. Streicher. *Domain-theorectic foundations of functional programs*. World Scientific. 2006.

Avaliação.

- Teste único.
- Assiduidade e participação nas aulas (até $|2|$ valores em torno da nota do teste.)

Aula 1. [(T) Seg, 31 Mar 2008, 11-13 h]

Sumário:
Apresentação da disciplina: motivação, programa, sistema de avaliação.
Sintaxe *vs* semântica. Noção de sintaxe abstracta.
Abordagens fundamentais à semântica das linguagens de programação *vs* áreas de aplicação típicas (*e.g.*, e respectivamente, *análise e cálculo dos programas*, *concepção de linguagens* e *verificação de programas*).



Mapa de linguagens a abordar neste curso

LIS	linguagem imperativa simples
LFS	linguagem funcional simples: ■ com *passagem de parâmetros por valor* (*call by value*) ■ com *passagem de parâmetros por referência* (*call by name*)
LFS++	linguagem funcional com tipos de ordem superior (produto e exponencial): ■ com avaliação *eager* ■ com avaliação *lazy*

Aula 2. [(T) Ter, 1 Abr 2008, 09-11 h]

Sumário:
Introdução à Semântica Operacional estrutural.
Exemplificação com uma pequena linguagem imperativa LIS.
Semântica *de transições* para LIS. Derivações. Propriedades.

LIS**: Sintaxe**

$$\begin{array}{lcl} P & ::= & C \mid E \mid B \\ C & ::= & \mathsf{skip} \mid l := E \mid C \,;\, C \mid \mathsf{if}\ B\ \mathsf{then}\ C\ \mathsf{else}\ C \mid \mathsf{while}\ B\ \mathsf{do}\ C \\ E & ::= & n \mid l \mid E\ iop\ E \\ B & ::= & b \mid l \mid E\ brel\ E \end{array}$$

com $n \in \mathbb{Z}$, $b \in \mathbb{B}$ e $l \in \{l_1, l_2, \cdots\}$

LIS**: Semântica operacional (dita *transacional, estrutural* ou *small-step*)**

$$(loc) \quad \langle l, s\rangle \longrightarrow \langle n, s\rangle \quad \Leftarrow\ l \in \mathrm{dom}\ s \wedge s\,l = n$$

$$(op1) \quad \frac{\langle E_1, s\rangle \longrightarrow \langle E_1', s'\rangle}{\langle E_1\ op\ E_2, s\rangle \longrightarrow \langle E_1'\ op\ E_2, s'\rangle}$$

$$(op2) \quad \frac{\langle E_2, s\rangle \longrightarrow \langle E_2', s'\rangle}{\langle n_1\ op\ E_2, s\rangle \longrightarrow \langle n_1\ op\ E_2', s'\rangle}$$

$$(op3) \quad \langle n_1\ op\ n_2, s\rangle \longrightarrow \langle c, s\rangle \quad \Leftarrow\ c = n_1\ op\ n_2$$

$$(set1) \quad \frac{\langle E, s\rangle \longrightarrow \langle E', s'\rangle}{\langle l := E, s\rangle \longrightarrow \langle l := E', s'\rangle}$$

$$(set2) \quad \langle l := n, s\rangle \longrightarrow \langle \mathsf{skip}, s[n/l]\rangle$$

$$(seq1) \quad \frac{\langle C_1, s\rangle \longrightarrow \langle C_1', s'\rangle}{\langle C_1\ ;\ C_2, s\rangle \longrightarrow \langle C_1'\ ;\ C_2, s'\rangle}$$

$$(seq2) \quad \langle \mathsf{skip}\ ;\ C, s\rangle \longrightarrow \langle C, s\rangle$$

$$(if1) \quad \frac{\langle B, s\rangle \longrightarrow \langle B', s'\rangle}{\langle \mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2, s\rangle \longrightarrow \langle \mathsf{if}\ B'\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2, s'\rangle}$$

$$(if2) \quad \langle \mathsf{if\ true\ then}\ C_1\ \mathsf{else}\ C_2, s\rangle \longrightarrow \langle C_1, s\rangle$$

$$(if3) \quad \langle \mathsf{if\ false\ then}\ C_1\ \mathsf{else}\ C_2, s\rangle \longrightarrow \langle C_2, s\rangle$$

$$(wh1) \quad \langle \mathsf{while}\ B\ \mathsf{do}\ C, s\rangle \longrightarrow \langle \mathsf{if}\ B\ \mathsf{then}\ (C\ ;\ \mathsf{while}\ B\ \mathsf{do}\ C)\ \mathsf{else\ skip}, s\rangle$$

Aula 3. [(TP) Seg, 7 Abr 2008, 11-13 h]

Sumário:
Os métodos indutivos como ferramenta de cálculo em Semântica. Revisão das noções de *indução matemática* e *indução estrutural*, como casos particulares de esquemas indutivos sobre *ordens bem-fundadas*. Resolução de exercícios.

- *Indução matemática*:

$$(\phi\, 0 \;\wedge\; \langle \forall\, n \;:\; n \in \mathbb{N} :\; \phi\, n \;\Rightarrow \phi\,(n+1)\rangle) \;\Rightarrow\; \langle \forall\, n \;:\; n \in \mathbb{N} :\; \phi\, n\rangle$$

- *Indução estrutural* (ie, sobre estruturas polinomiais ('árvores') finitas):

$$(\langle \forall\, a \;:\; a \in A :\; \phi\, a\rangle \;\wedge\; \langle \forall\, c \;:\; c \in C :\; \phi\, t_1 \cdots \phi\, t_n \;\Rightarrow \phi\, c(t_1, \cdots t_n)\rangle) \Rightarrow \langle \forall\, t \;:\; t \in T :\; \phi\, t\rangle$$

  onde A e C designam os conjuntos de construtores com 1 e mais que 1 argumentos, respectivamente.

- *Indução noetheriana* (ie. sobre uma *ordem bem-fundada* $\langle U, \prec\rangle$):

$$(\langle \forall\, u \;:\; u \in U :\; \langle \forall\, v \;:\; v \prec u :\; \phi\, v\rangle \Rightarrow \phi\, u)\rangle \;\Rightarrow\; \langle \forall\, u \;:\; u \in U :\; \phi\, u\rangle$$

Exercício. Mostre que $\phi\, n = n^2$, para $\phi\, n \triangleq \sum_{j=1}^{n}(2j-1)$.

Exercício. Mostre que não existe nenhuma string u tal que, para símbolos a e b distintos, $au = ub$.

Exercício. Mostre por indução estrutural que

$$len \cdot (map\, f) \;=\; len$$

Exercício. Formule e prove um teorema similar sobre árvores binárias.

Exercício. Formule o esquema indutivo estrutural sobre árvores generalizadas (ie, com um número finito mas arbitrário de descendentes).

Aula 4. [(T) Ter, 8 Abr 2008, 09-11 h]

Sumário:
Definições indutivas por conjuntos de *regras*. Esquema indutivo associado.

- *Indução sobre a estrutura de* (A, R):

$$(\langle \forall\, a \;:\; a \in A :\; \phi\, a \rangle \;\wedge\; \langle \forall\, r \;:\; r = [(h_1, \cdots, h_n) \vdash c] \in R :\; \phi\, h_1 \cdots \phi\, h_n \;\Rightarrow \phi\, c \rangle) \;\Rightarrow\; \langle \forall\, t \;:\; t \in TD :\; \phi\, t \rangle$$

  onde A e R designam os conjuntos de *axiomas* e *regras de construção*, respectivamente, que definem indutivamente um (sub-)conjunto de termos TD.

Exercício. Demonstre que o sistema de regras de transição usado na aula anterior para especificar a semântica operacional da linguagem LIS é determinístico. Para isso, prove por indução sobre a estrutura das regras

$$\Psi(P, s, P', s') \;\triangleq\; \langle P, s \rangle \longrightarrow \langle P', s' \rangle \;\wedge\; \langle \forall\, P'', s'' \;:\; : \; (\langle P, s \rangle \longrightarrow \langle P'', s'' \rangle) \;\Rightarrow\; (P' = P'' \wedge s' = s'') \rangle$$

Resolução (parcial). Consideremos apenas o fecho para a regra $(set1)$:

$$\Psi(E, s, E', s') \;\Rightarrow\; \Psi(l := E, s, l := E', s')$$

- que a transição $\langle l := E, s \rangle \longrightarrow \langle l := E', s' \rangle$ decorre directamente de $(set1)$
- resta mostrar que

$$\langle l := E, s \rangle \longrightarrow \langle P'', s'' \rangle \;\;\Rightarrow\;\; P'' = l := E' \;\wedge\; s'' = s'$$

  A única regra aplicável (sem violar a HI $\Psi(E, s, E', s')$) é $(set1)$. Note-se que, por exemplo, a aplicação de $(set2)$ obrigaria E a ser um inteiro e violaria o primeiro factor conjuntivo da HI. Assim, por $(set1)$, vem $P'' = l := E''$ para algum E'' tal que

$$\langle E, s \rangle \longrightarrow \langle E'', s'' \rangle$$

  mas a HI implica que $E' = E''$ e $s' = s''$. Logo $P'' = l := E'' = l := E'$ e $s' = s''$.

Questão. A que conclusão se chegaria se o predicado Ψ fosse relaxado para

$$\Psi'(P, s, P', s') \;\triangleq\; \langle \forall\, P'', s'' \;:\; : \; (\langle P, s \rangle \longrightarrow \langle P'', s'' \rangle) \;\Rightarrow\; (P' = P'' \wedge s' = s'') \rangle$$

?

Aula 5. [(T) Seg, 14 Abr 2008, 11-13 h]

Sumário:
Semântica de *avaliação* para LIS. Equivalência entre as semânticas de *transição* e de *avaliação*.

LIS**: Semântica operacional (dita *de avaliação, natural* ou *big-step*)**

$$(EVcon) \qquad \langle c, s\rangle \Downarrow \langle c, s\rangle \quad (c \in \mathbb{Z} \cup \mathbb{B})$$

$$(EVloc) \qquad \langle l, s\rangle \Downarrow \langle n, s\rangle \quad \Leftarrow\ l \in \mathit{dom}\, s\ \wedge\ s\, l = n$$

$$(EVop) \qquad \frac{\langle E_1, s\rangle \Downarrow \langle n_1, s'\rangle \quad \langle E_2, s'\rangle \Downarrow \langle n_2, s''\rangle}{\langle E_1\, op\, E_2, s\rangle \Downarrow \langle c, s''\rangle} \quad \Leftarrow\ c = n_1 op\, n_2$$

$$(EVskip) \qquad \langle \mathsf{skip}, s\rangle \Downarrow \langle \mathsf{skip}, s\rangle$$

$$(EVset) \qquad \frac{\langle E, s\rangle \Downarrow \langle n, s'\rangle}{\langle l := E, s\rangle \Downarrow \langle \mathsf{skip}, s'[n/l]\rangle}$$

$$(EVseq) \qquad \frac{\langle C_1, s\rangle \Downarrow \langle \mathsf{skip}, s'\rangle \quad \langle C_2, s'\rangle \Downarrow \langle \mathsf{skip}, s''\rangle}{\langle C_1\ ;\ C_2, s\rangle \Downarrow \langle \mathsf{skip}, s''\rangle}$$

$$(EVif1) \qquad \frac{\langle B, s\rangle \Downarrow \langle \mathsf{true}, s'\rangle \quad \langle C_1, s'\rangle \Downarrow \langle \mathsf{skip}, s''\rangle}{\langle \mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2, s\rangle \Downarrow \langle \mathsf{skip}, s''\rangle}$$

$$(EVif2) \qquad \frac{\langle B, s\rangle \Downarrow \langle \mathsf{false}, s'\rangle \quad \langle C_2, s'\rangle \Downarrow \langle \mathsf{skip}, s''\rangle}{\langle \mathsf{if}\ B\ \mathsf{then}\ C_1\ \mathsf{else}\ C_2, s\rangle \Downarrow \langle \mathsf{skip}, s''\rangle}$$

$$(EVwh1) \qquad \frac{\langle B, s\rangle \Downarrow \langle \mathsf{true}, s'\rangle \quad \langle C, s'\rangle \Downarrow \langle \mathsf{skip}, s''\rangle \quad \langle \mathsf{while}\ B\ \mathsf{do}\ C, s''\rangle \Downarrow \langle \mathsf{skip}, s'''\rangle}{\langle \mathsf{while}\ B\ \mathsf{do}\ C, s\rangle \Downarrow \langle \mathsf{skip}, s'''\rangle}$$

$$(EVwh2) \qquad \frac{\langle B, s\rangle \Downarrow \langle \mathsf{false}, s'\rangle}{\langle \mathsf{while}\ B\ \mathsf{do}\ C, s\rangle \Downarrow \langle \mathsf{skip}, s'\rangle}$$

Teorema (equivalência entre $\longrightarrow$ e $\Downarrow$) Para toda a configuração $\langle P, s\rangle$ e toda a configuração terminal $\langle V, s'\rangle$, no contexto da linguagem LIS, tem-se que

$$\langle P, s\rangle \Downarrow \langle V, s'\rangle \ \Leftrightarrow\ \langle P, s\rangle \longrightarrow^* \langle V, s'\rangle \tag{1}$$

onde $\longrightarrow^*$ designa o fecho reflexivo e transitivo da semântica de transição $\longrightarrow$.

Teorema (propriedades de $\Downarrow$) Para toda a configuração $\langle P, s\rangle$ e todas as configurações terminais $\langle V, s'\rangle$ e $\langle V', s''\rangle$ no contexto da linguagem LIS, tem-se

- Se $\langle P, s\rangle \Downarrow \langle V, s'\rangle$ e $\langle P, s\rangle \Downarrow \langle V', s''\rangle$, então $V = V'$ e $s' = s''$, i.e., a semântica de avaliação é *determinística*.
- Se $\langle P, s\rangle \Downarrow \langle V, s'\rangle$, então as expressões P e V são do mesmo tipo (i.e., da mesma categoria sintática).
- Se $\langle P, s\rangle \Downarrow \langle V, s'\rangle$ e P é uma expressão bolleana ou inteira, então $s = s'$, i.e. a avaliação de expressões não produz *efeitos laterais* no estado.

Exercício. Mostre que

$$\langle \mathsf{while}\, l > 0\, \mathsf{do}\, l := 0, \{l \to 1\}\rangle \Downarrow \langle \mathsf{skip}, s''\rangle$$

para algum s''

Resolução. Façamos, para simplificar a escrita, $D = \mathsf{while}\, l > 0\, \mathsf{do}\, l := 0$ e $s = \{l \to 1\}$. Comecemos por notar que a guarda do ciclo avalia em $\langle \mathsf{true}, s\rangle$, porque

$$T_1 \;=\; \frac{\dfrac{}{\langle l, s\rangle \Downarrow \langle 1, s\rangle}\,(EVloc) \qquad \dfrac{}{\langle 0, s\rangle \Downarrow \langle 0, s\rangle}\,(EVcon)}{\langle l > 0, s\rangle \Downarrow \langle \mathsf{true}, s\rangle}\,(EVop)$$

Então a regra a aplicar terá de ser $(EVwh1)$ (porquê?):

$$\frac{T_1 \quad \dfrac{\vdots}{\langle l := 0, s\rangle \Downarrow \langle \mathsf{skip}, s'\rangle}\,(EVset) \qquad \dfrac{\vdots}{\langle D, s'\rangle \Downarrow \langle \mathsf{skip}, s''\rangle}\,(EVwh2)}{\langle D, s\rangle \Downarrow \langle \mathsf{skip}, s''\rangle}\,(EVwh1)$$

para algum s' e s''. A hipótese do meio é deduzida via $(EVset)$ na árvore seguinte

$$T_2 \;=\; \frac{\dfrac{}{\langle 0, s\rangle \Downarrow \langle 0, s\rangle}\,(EVcon)}{\langle l := 0, s\rangle \Downarrow \langle \mathsf{skip}, s'\rangle}\,(EVset)$$

com $s' = \{l \to 0\}$. Finalmente, a última prova origina

$$T_3 \;=\; \frac{\dfrac{\dfrac{}{\langle l, s'\rangle \Downarrow \langle 0, s'\rangle}\,(EVloc) \qquad \dfrac{}{\langle 0, s'\rangle \Downarrow \langle 0, s'\rangle}\,(EVcon)}{\langle l > 0, s'\rangle \Downarrow \langle \mathsf{false}, s'\rangle}\,(EVop)}{\langle D, s\rangle \Downarrow \langle \mathsf{skip}, s'\rangle}\,(EVwh2)$$

Note-se que $s'' = s'$. Finalmente,

$$\frac{T_1 \quad T_2 \quad T_3}{\langle D, s\rangle \Downarrow \langle \mathsf{skip}, s'\rangle}\,(EVwh1)$$

Aula 6. [(TP) Ter, 15 Abr 2008, 09-11 h]

Sumário:
Não houve por impedimento do docente.

Aula 7. [(TP) Seg, 21 Abr 2008, 11-13 h]

Sumário:
Resolução de exercícios sobre Semântica Operacional.

Exercício. Discuta a adequação da seguinte regra semântica que alguém propôs para captar o significado operacional do comando $\mathsf{while}\, B\, \mathsf{do}\, C$

$$\frac{\langle B, s\rangle \longrightarrow \langle B', s\rangle}{\langle \mathsf{while}\, B\, \mathsf{do}\, C, s\rangle \longrightarrow \langle \mathsf{while}\, B'\, \mathsf{do}\, C, s'\rangle}$$

Exercício. Mostre que não existe nenhum s' tal que

$$\langle \mathsf{while\ true\ do\ skip}, s\rangle \Downarrow \langle \mathsf{skip}, s'\rangle$$

qualquer que seja o estado inicial s.

Exercício. Considere o seguinte programa na linguagem LIS:

$$P \triangleq y := x; a := 1; \mathsf{while}\, y > 0\, \mathsf{do}\, (a := a * y; y := y - 1)$$

Calcule uma configuração terminal de $\langle P, s\rangle$, para $s = \{x \to 3, y \to 2, a \to 9\}$, mostrando que

$$\langle P, s\rangle \longrightarrow^* \langle \mathsf{skip}, s'\rangle$$

Qual o valor de a em s'?

Exercício. Reporte-se à semântica natural para a linguagem LIS introduzida nas aulas. Mostre que para toda a expressão inteira E existe um número $n \in \mathbb{Z}$ tal que

$$\langle E, s\rangle \Downarrow \langle n, s\rangle$$

qualquer que seja o estado de avaliação s. Esta propriedade é designada por *normalização*.

Exercício. Formalize e prove a seguinte propriedade: *a avaliação de expressões inteiras na linguagem* LIS *é determinística*.

Exercício. Formalize e prove um resultado análogo ao do exercício anterior para a semântica de transições de LIS.

Exercício. Complete as provas dos teoremas enunciados no sumário da aula anterior (continuando o trabalho feito na aula teórica).

Aula 8. [(TP) Seg, 22 Abr 2008, 9-11 h]

Sumário:
Equivalência semântica. Resolução de exercícios sobre equivalência semântica.

Equivalência semântica

$$P_1 \cong P_2 \;\Leftrightarrow\; \langle \forall\, V, s' \;::\; \langle P_1, s\rangle \Downarrow \langle V, s'\rangle \;\Leftrightarrow\; \langle P_2, s\rangle \Downarrow \langle V, s'\rangle\rangle$$

Teorema (congruência)
A relação $\cong$ é uma congruência sobre configurações.

Exercício. Mostre que

- $\mathsf{if}\ B\ \mathsf{then}\ C\ \mathsf{else}\ C'; C''' \;\cong\; \mathsf{if}\ B\ \mathsf{then}\ C; C''\ \mathsf{else}\ C'; C''$
- $C \;\cong\; \mathsf{skip}; C \;\cong\; C; \mathsf{skip}$
- $\mathsf{if\ true\ then}\ C\ \mathsf{else}\ C' \;\cong\; C$

Exercício. Considere a seguinte regra semântica para a introdução de um comando que permite efectuar declarações locais de referências:

$$\frac{\langle E, s\rangle \Downarrow \langle n, s'\rangle \qquad \langle C[l'/l], s'[n/l']\rangle \Downarrow \langle \mathsf{skip}, s''[n'/l']\rangle}{\langle \mathsf{local}\, l := E; C\ \mathsf{endlocal}, s\rangle \Downarrow \langle \mathsf{skip}, s''\rangle}$$

Mostre que

$$\langle C, s\rangle \Downarrow \langle \mathsf{skip}, s[s(l_2)/l_1, s(l_1)/l_2]\rangle$$

para $C \triangleq \mathsf{local}\, l := l_1; l_1 := l_2; l_2 := l\ \mathsf{endlocal}$.

Exercício. Nas condições do exercício anterior, mostre que, para $x \neq y$ se tem

$$\mathsf{local}\, x := E; y := x\ \mathsf{endlocal} \;\cong\; y := E$$

Permanecerá válida esta equivalência caso $x = y$?

Aula 9. [(T) Seg, 28 Abr 2008, 11-13 h]

Sumário:
Introdução à semântica denotacional.

Semântica denotacional
O significado dos programas é dado pelo seu mapeamento numa estrutura matemática adequada. A êmfase é colocada na construção de *modelos* matemáticos para as liguagens de programação:

- a cada componente C de um programa é dada uma *denotação* $[\![C]\!]$, ie um elemento de uma estrutura matemática que representa o significado de C em *qualquer* programa em que ocorra;
- a denotação de uma componente é totalmente determinada pelas denotações das suas sub-componentes, ie a semântica é *composicional*

Exemplo: expressões aritméticas.

$$[\![\,]\!] : E \longrightarrow \mathbb{Z}$$

com, por exemplo,

$$\begin{aligned} [\![n]\!] &= n \\ [\![E{+}F]\!] &= [\![E]\!] + [\![F]\!] \end{aligned}$$

Exercício. Calcule $[\![(4 + (3 + 9))]\!]$ e mostre que em qualquer expressão aritmética na linguagem LIS a colocação de parentesis é arbitrária.

Denotações de programas como funções parciais
Seja Σ a denotação matemática de uma noção conveniente de estado para a linguagem LIS. Então,

$$\begin{aligned} [\![\,]\!] &: C \longrightarrow \Sigma \rightharpoonup \Sigma \\ [\![\,]\!] &: B \longrightarrow \Sigma \rightharpoonup \mathbb{B} \\ [\![\,]\!] &: E \longrightarrow \Sigma \rightharpoonup \mathbb{Z} \end{aligned}$$

$$\begin{aligned} [\![\mathsf{skip}]\!] &= \mathsf{id} \\ [\![C\,;C']\!] &= [\![C']\!] \cdot [\![C]\!] \\ [\![x]\!] &= \lambda\, s \in \Sigma\,.\ \langle x \in \mathit{dom}\, s \ \rightarrow\ s\, x\rangle \\ [\![l := E]\!] &= \lambda\, s \in \Sigma\,.\ \langle s \in \mathit{dom}\, [\![E]\!] \ \rightarrow\ s[[\![E]\!]/l]\rangle \\ [\![\mathsf{if}\ B\ \mathsf{then}\ C\ \mathsf{else}\ C']\!] &= \lambda\, s \in \Sigma\,.\ \langle [\![B]\!]\, s \ \rightarrow\ [\![C]\!]\, s,\ [\![C']\!]\, s\rangle \\ \\ [\![\mathsf{while}\ B\ \mathsf{do}\ C]\!] &= ??? \end{aligned}$$

Aula 10. [(T) Ter, 29 Abr 2008, 9-11 h]

Sumário:
Noção de domínio. Monotonia, continuidade e estriticidade.

Motivação.
De que forma podemos construir a denotação do programa $\mathsf{while}\, x > 0\, \mathsf{do}\, (y := x * y; x := x - 1)$? Como o seu estado se resume a duas variáveis, consideremos $\Sigma = Z \times Z$. Se recorrermos, como habitualmente, à semântica operacional para definir o significado ciclo $\mathsf{while}\, x > 0\, \mathsf{do}\, (y := x * y; x := x - 1)$ concluimos que este deverá verificar a propriedade seguinte:

$$
\begin{array}{ll}
 & [\![\mathsf{while}\, x > 0\, \mathsf{do}\, (y := x * y; x := x - 1)]\!] \\
= & \quad \{\ \text{semântica operacional}\ \} \\
 & [\![\mathsf{if}\, x > 0\, \mathsf{then}\, y := x * y; x := x - 1; \mathsf{while}\, x > 0\, \mathsf{do}\, (y := x * y; x := x - 1)\, \mathsf{else}\, \mathsf{skip}]\!] \\
= & \quad \{\ \text{semântica denotacional do condicional}\ \} \\
 & \lambda(x,y).\ \begin{cases} [\![\mathsf{skip}]\!](x,y) & \Leftarrow\ x \leq 0 \\ [\![y := x * y; x := x - 1; \mathsf{while}\, x > 0\, \mathsf{do}\, (y := x * y; x := x - 1)]\!](x,y) & \Leftarrow\ x > 0 \end{cases} \\
= & \quad \{\ \text{semântica denotacional de skip e da composição sequencial}\ \} \\
 & \lambda(x,y).\ \begin{cases} (x,y) & \Leftarrow\ x \leq 0 \\ [\![\mathsf{while}\, x > 0\, \mathsf{do}\, (y := x * y; x := x - 1)]\!] \cdot [\![y := x * y; x := x - 1]\!](x,y) & \Leftarrow\ x > 0 \end{cases} \\
= & \quad \{\ \text{semântica denotacional da atribuição}\ \} \\
 & \lambda(x,y).\ \begin{cases} (x,y) & \Leftarrow\ x \leq 0 \\ [\![\mathsf{while}\, x > 0\, \mathsf{do}\, (y := x * y; x := x - 1)]\!](x-1, x*y) & \Leftarrow\ x > 0 \end{cases}
\end{array}
$$

Ficamos, pois, com uma equação em $\omega = [\![\mathsf{while}\, x > 0\, \mathsf{do}\, (y := x * y; x := x - 1)]\!]$:

$$\omega = f\, \omega \tag{2}$$

Não se trata de uma equação numa estrutura familiar e rica em propriedades (como, por exemplo, o corpo dos reais) pelo que necessitaremos de caracterizar a estrutura em que se define e procurar métodos para a resolver, determinando a denotação do ciclo.

Domínio
Uma ordem parcial completa $(D, \sqsubseteq)$ é uma ordem parcial onde todas as cadeias crescente contáveis $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \sqsubseteq \ldots$ têm supremo (que representamos por $\langle \bigsqcup\ n\ :\ n \geq 0 :\ d_n \rangle$), i.e.

$$\langle \forall\ k\ :\ k \geq 0 :\ d_k \sqsubseteq \langle \bigsqcup\ n\ :\ n \geq 0 :\ d_n \rangle \rangle \tag{3}$$

$$\langle \forall\ x\ :\ x \in D :\ (\langle \forall\ k\ :\ k \geq 0 :\ d_k \sqsubseteq x \rangle) \Rightarrow \langle \bigsqcup\ n\ :\ n \geq 0 :\ d_n \rangle \sqsubseteq x \rangle \tag{4}$$

Se, adicionalmente, $(D, \sqsubseteq)$ possuir elemento mínimo, i.e.

$$\langle \exists\ \bot\ :\ \bot \in D :\ \langle \forall\ x\ :\ x \in D :\ \bot \sqsubseteq x \rangle \rangle \tag{5}$$

será classificado como um domínio.

Monotonia, continuidade e estriticidade
Uma função $f : (D_1, \sqsubseteq_1, \bot_1) \longrightarrow (D_2, \sqsubseteq_2, \bot_2)$ entre domínios será

monótona se preservar a ordem, i.e., $x \sqsubseteq_1 y \;\Rightarrow\; f\,x \sqsubseteq_2 f\,y$ (6)

contínua se preservar supremos, i.e., $f(\langle \bigsqcup\; n \;:\; n \geq 0 :\; d_n \rangle) = \langle \bigsqcup\; n \;:\; n \geq 0 :\; f\,d_n \rangle$ (7)

estrita se preservar o mínimo, i.e., $f\,\bot_1 = \bot_2$ (8)

Exemplo: domínios definidos sobre os números naturais

Exercício.

O domínio das funções parciais $A \rightharpoonup B$

ordem: $f \sqsubseteq g \;\Leftrightarrow\; (\mathrm{dom}\; f \subseteq \mathrm{dom}\; g \wedge \langle \forall\, x \;:\; x \in \mathrm{dom}\; f :\; f\,x = g\,x \rangle)$

mínimo: $\bot$ é a função totalmente indefinida, i.e., $\mathrm{dom}\; \bot = \emptyset$

supremo: o supremo de uma cadeia contável $f_1 \sqsubseteq f_2 \sqsubseteq f_3 \sqsubseteq ...$ é uma função f tal que

- $\mathrm{dom}\; f = \langle \bigcup\; n \;:\; n \geq 0 :\; \mathrm{dom}\; f_n \rangle$
- $f\,x = \Big\{ f_n\,x \quad \Leftarrow \quad \langle \exists\, n \;:\; n \geq 0 :\; x \in \mathrm{dom}\; f_n \rangle$

- Mostre que a definição acima define efectivamente um domínio.

Exercício. Considere a função f na equação (2). Indique o seu tipo e mostre que se trata de uma função contínua.

Exercício. Considere um ordem parcial qualquer sobre um conjunto finito de elementos. Que condições deve esta satisfazer para poder ser classificada como domínio? Justifique.

Exercício. Mostre que a função identidade num domínio é contínua e que a composta de duas funções contínuas é ainda contínua.

Exercício. Justifique a afirmação seguinte: *para mostrar a continuidade de uma função monótona, é suficiente que para qualquer cadeia ascendente contável se tenha*

$$f(\langle \bigsqcup\; n \;:\; n \geq 0 :\; d_n \rangle) \sqsubseteq \langle \bigsqcup\; n \;:\; n \geq 0 :\; f\,d_n \rangle$$

Exercício. Seja $(\mathbb{N} \cup \{\infty\}, \leq)$ a ordem usual nos naturais estendida de forma a $\langle \forall\, n \;:\; n \in \mathbb{N} :\; n \leq \infty \rangle$. Mostre que a função $f : \mathbb{N} \cup \{\infty\} \longrightarrow \mathbb{N} \cup \{\infty\}$ dada por

$$\begin{cases} f\,n = 0 \quad \Leftarrow\; n \in \mathbb{N} \\ f\,\infty = \infty \end{cases}$$

é monótona e estrita mas não contínua.

Aula 11. [(TP) Seg, 19 Mai 2008, 11-13 h]

Sumário:
Pontos fixos de uma função. Cadeias de Kleene e sua utilização no cálculo de pontos fixos de funções sobre domínios. Cálculo da semântica dos ciclos while.

Pontos fixos
Seja $f : D \longrightarrow D$ uma função sobre uma ordem parcial $(D, \sqsubseteq)$ é um elemento $x \in D$. Então x diz-se

- um *ponto fixo* de f se $x = f\, x$
- um *pre-ponto fixo* de f se $f\, x \sqsubseteq x$
- um *post-ponto fixo* de f se $x \sqsubseteq f\, x$

Teorema
O menor pre-ponto fixo de uma função monótona $f : D \longrightarrow D$ sobre uma ordem parcial D é simultaneamente o menor dos seus pontos fixos.

Prova

$$
\begin{array}{ll}
 & \mu_f \text{ é o menor pre-ponto fixo de } f \\
\Leftrightarrow & \quad \{ \text{ definição de menor pre-ponto fixo } \} \\
 & f\,\mu_f \sqsubseteq \mu_f \ \wedge\ \langle \forall\, x \ :\ x \in D :\ f\, x \sqsubseteq x \Rightarrow \mu_f \sqsubseteq x \rangle \\
\Rightarrow & \quad \{ \ f \text{ é monótona } \} \\
 & f\,\mu_f \sqsubseteq \mu_f \ \wedge\ f\, f\,\mu_f \sqsubseteq f\,\mu_f \ \wedge\ \langle \forall\, x \ :\ x \in D :\ f\, x \sqsubseteq x \Rightarrow \mu_f \sqsubseteq x \rangle \\
\Rightarrow & \quad \{ \ f\,\mu_f \text{ é pré-ponto fixo de } f \text{ e } \mu_f \text{ é o menor de entre estes } \} \\
 & \mu_f \sqsubseteq f\,\mu_f \ \wedge\ f\,\mu_f \sqsubseteq \mu_f \ \wedge\ f\, f\,\mu_f \sqsubseteq f\,\mu_f \ \wedge\ \langle \forall\, x \ :\ x \in D :\ f\, x \sqsubseteq x \Rightarrow \mu_f \sqsubseteq x \rangle \\
\Rightarrow & \quad \{ \text{ eliminação da conjunção (i.e., } \phi \wedge \psi \Rightarrow \phi) \ \} \\
 & \mu_f \sqsubseteq f\,\mu_f \ \wedge\ f\,\mu_f \sqsubseteq \mu_f \\
\Leftrightarrow & \quad \{ \text{ anti-simetria de } \sqsubseteq \} \\
 & \mu_f = f\,\mu_f
\end{array}
$$

Teorema (de Knaster-Tarski, 1955)
Seja $(D, \sqsubseteq)$ um recticulado completo e $f : D \longrightarrow D$ uma função monótona Então, o conjunto dos pontos fixos de f é não vazio e forma um recticulado completo. Em particular, a base e o topo desse recticulado (i.e., o menor e o maior ponto fixo de f) são calculados por

$$
\begin{aligned}
\mu_f &= \bigsqcap \{x \in D |\ f(x) \sqsubseteq x\} \\
\nu_f &= \bigsqcup \{x \in D |\ x \sqsubseteq f(x)\}
\end{aligned}
$$

Prova (de que μ_f é o menor ponto fixo de f; o resto fica como exercício)

Seja $P = \{x \in D |\ f(x) \sqsubseteq x\}$. Sendo D um recticulado completo, $\mu_f = \bigsqcap P$ existe em D. Então,

$$
\begin{array}{ll}
 & \langle \forall\, x \ :\ x \in P :\ \mu_f \sqsubseteq x \rangle \\
\Rightarrow & \quad \{\ f \text{ é monótona}\ \} \\
 & \langle \forall\, x \ :\ x \in P :\ f\, \mu_f \sqsubseteq f\, x \rangle \\
\Rightarrow & \quad \{\ \text{definição de } P \text{ e transitividade}\ \} \\
 & \langle \forall\, x \ :\ x \in P :\ f\, \mu_f \sqsubseteq x \rangle \quad \text{(i.e., } f\, \mu_f \text{ é um minorante de } P\text{)} \\
\Rightarrow & \quad \{\ \text{por definição, } \mu_f \text{ é o maior dos minorantes de } P\ \} \\
 & f\, \mu_f \sqsubseteq \mu_f \\
\Rightarrow & \quad \{\ \text{por definição de } P\ \} \\
 & \mu_f \in P \quad \text{(i.e., } \mu_f \text{ é o menor pre-ponto fixo de } f\text{)} \\
\Rightarrow & \quad \{\ \text{pelo resultado anterior}\ \} \\
 & \mu_f \text{ é o menor ponto fixo de } f
\end{array}
$$

O Teorema de Kleene
Seja $f : D \longrightarrow D$ uma função contínua sobre um domínio D. Então o menor ponto fixo de f existe e é calculado por

$$\mu_f \ =\ \langle \bigsqcup\ n \ :\ n > 0 :\ f^n \perp \rangle \tag{9}$$

onde

$$\begin{cases} f^0 \perp & \triangleq\ \perp \\ f^{n+1} \perp & \triangleq\ f\, f^n \perp \end{cases}$$

Prova
Verifiquemos que μ_f definido pela equação (9) é ponto fixo de f (o resto da prova sugere-se como exercício)

$$
\begin{array}{ll}
 & f\, \mu_f \\
= & \quad \{\ \text{definição de } \mu_f\ \} \\
 & f\, \langle \bigsqcup\ n \ :\ n > 0 :\ f^n \perp \rangle \\
= & \quad \{\ f \text{ é contínua}\ \} \\
 & \langle \bigsqcup\ n \ :\ n > 0 :\ f\, f^n \perp \rangle \\
= & \quad \{\ \text{definição de } f^n\ \} \\
 & \langle \bigsqcup\ n \ :\ n > 0 :\ f^{n+1} \perp \rangle \\
= & \quad \{\ \text{supremo de cadeia indexada por } \mathbb{N}\ \} \\
 & \mu_f
\end{array}
$$

Exercício. Complete a prova do teorema de Kleene, mostrando, por indução sobre n, que μ_f definido pela equação (9) é o menor ponto fixo de f.

Exemplo. Resolvamos, agora, a equação (2) por recurso ao teorema de Kleene (sobre que domínio?), i.e.,

$$\omega = f\, \omega$$

onde $f : \Sigma \longrightarrow \Sigma$ é a função

$$f\,\omega\,(x,y) = \begin{cases} (x,y) & \Leftarrow \quad x \leq 0 \\ \omega(x-1, x*y) & \Leftarrow \quad x > 0 \end{cases}$$

Calculemos, então, uma solução para a equação (2) como limite de uma cadeia de aproximações:

$$\begin{cases} \omega_0 = \bot \\ \omega_{n+1} = f\,\omega_n \end{cases}$$

Assim,

$$\begin{aligned}
\omega_0 =& \bot \\
\omega_1 =& \lambda(x,y).\ \begin{cases} (x,y) & \Leftarrow \quad x \leq 0 \\ \bot & \Leftarrow \quad x > 0 \end{cases} \\
\omega_2 =& \lambda(x,y).\ \begin{cases} (x,y) & \Leftarrow \quad x \leq 0 \\ (0,y) & \Leftarrow \quad x = 1 \\ \bot & \Leftarrow \quad x \geq 2 \end{cases} \\
\omega_3 =& \lambda(x,y).\ \begin{cases} (x,y) & \Leftarrow \quad x \leq 0 \\ (0,y) & \Leftarrow \quad x = 1 \\ (0,2*y) & \Leftarrow \quad x = 2 \\ \bot & \Leftarrow \quad x \geq 3 \end{cases} \\
\omega_4 =& \lambda(x,y).\ \begin{cases} (x,y) & \Leftarrow \quad x \leq 0 \\ (0,y) & \Leftarrow \quad x = 1 \\ (0,2*y) & \Leftarrow \quad x = 2 \\ (0,6*y) & \Leftarrow \quad x = 3 \\ \bot & \Leftarrow \quad x \geq 4 \end{cases} \\
& \ldots \\
\omega_n =& \lambda(x,y).\ \begin{cases} (x,y) & \Leftarrow \quad x \leq 0 \\ (0,x!*y) & \Leftarrow \quad 0 < x < n \\ \bot & \Leftarrow \quad x \geq n \end{cases}
\end{aligned}$$

O supremo desta cadeia de aproximações $\omega_0 \sqsubseteq \omega_1 \sqsubseteq \omega_2 \sqsubseteq \ldots$ é

$$\omega_\infty = \langle \bigsqcup\ n\ :\ n \geq 0 :\ \omega_n \rangle = \begin{cases} (x,y) & \Leftarrow \quad x \leq 0 \\ (0,x!*y) & \Leftarrow \quad x > 0 \end{cases} \tag{10}$$

é, efectivamente, solução da equação (2):

$$\begin{aligned}
f\,\omega_\infty \ &= \lambda(x,y).\ \begin{cases} (x,y) & \Leftarrow \quad x \leq 0 \\ \omega_\infty(x-1, x*y) & \Leftarrow \quad x > 0 \end{cases} \\
&= \ \lambda(x,y).\ \begin{cases} (x,y) & \Leftarrow \quad x \leq 0 \\ (0,1*y) & \Leftarrow \quad x = 1 \\ (0,(x-1)!*y*x) & \Leftarrow \quad x > 0 \end{cases} \\
&= \ \omega_\infty
\end{aligned}$$

Aula 12. [(TP) Ter, 20 Mai 2008, 9-11 h]

Sumário:
Resolução de exercícios: aplicações do teorema de Kleene e construções sobre domínios.

Exercício. Recorrendo ao teorema de Kleene determine e caracterize a menor solução da equação

$$x \;=\; \mathsf{id}_{\mathbb{N}} \cup r \cup x^{\circ} \cup x \cdot x$$

onde r é uma relação binária em $\mathbb{N}$. Caracterize a domínio que considerou e prove a continuidade da função da qual x é solução.

Exercício. Repita o exercício anterior para a seguinte equação sobre conjuntos

$$X \;=\; 1 + A \times X$$

Nota:
Em vários exercícios abaixo recorreremos ao seguinte resultado, facilmente verificável, e ao qual nos iremos referir como *troca*. Considere uma cadeia ascendente $x_{i,j}$ num domínio D, duplamente indexada por números naturais, e tal que

$$(i \leq i' \;\wedge\; j \leq j') \;\Rightarrow\; x_{i,j} \sqsubseteq x_{i',j'}$$

então,

$$\langle \bigsqcup\, i \;:\; i \geq 0 :\; \langle \bigsqcup\, j \;:\; j \geq 0 :\; x_{i,j} \rangle \rangle \;=\; \langle \bigsqcup\, k \;:\; k \geq 0 :\; x_{k,k} \rangle \;=\; \langle \bigsqcup\, j \;:\; j \geq 0 :\; \langle \bigsqcup\, i \;:\; i \geq 0 :\; x_{i,j} \rangle \rangle$$

Exercício. Mostre que uma função de dois argumentos $f : D_1 \times D_2 \longrightarrow E$, entre domínios, é monótona se o fôr em cada um dos argumentos.

Exercício. Mostre que uma função de dois argumentos $f : D_1 \times D_2 \longrightarrow D_3$ entre domínios, é contínua se preservar supremos de cadeias ascendentes contáveis em cada um dos argumentos separadamente, i.e.,

$$f\,(\langle \bigsqcup\, n \;:\; n \geq 0 :\; x_n \rangle, y) \;=\; \langle \bigsqcup\, n \;:\; n \geq 0 :\; f\,(x_n, y) \rangle$$
$$f\,(x, \langle \bigsqcup\, n \;:\; n \geq 0 :\; y_n \rangle) \;=\; \langle \bigsqcup\, n \;:\; n \geq 0 :\; f\,(x, y_n) \rangle$$

Exercício (domínio produto).

Domínio produto
Dados dois domínios $(D_1, \sqsubseteq_1, \bot_1, \bigsqcup_1)$ e $(D_2, \sqsubseteq_2, \bot_2, \bigsqcup_2)$, define-se um novo domínio (dito o *produto* de D_1 e D_2) tomando como portador o produto cartesiano $D_1 \times D_2$ e definindo

ordem: $(x, y) \sqsubseteq (x', y') \;\triangleq\; x \sqsubseteq_1 x' \wedge y \sqsubseteq_2 y'$

mínimo: $\bot = (\bot_1, \bot_2)$

supremo: $\langle \bigsqcup\; n \;:\; n \geq 0 :\; (x_n, y_n) \rangle \triangleq (\langle \bigsqcup_1\, i \;:\; i \geq 0 :\; x_i \rangle, \langle \bigsqcup_2\, j \;:\; j \geq 0 :\; y_j \rangle)$

- Mostre que a definição acima define efectivamente um domínio.
- Mostre que as funções canónicas de projecção, π_1 e π_2, são contínuas.

Exercício (domínio exponencial).

Domínio exponencial
Dados dois domínios $(D_1, \sqsubseteq_1, \bot_1, \bigsqcup_1)$ e $(D_2, \sqsubseteq_2, \bot_2, \bigsqcup_2)$, define-se um novo domínio (dito o domínio das funções contínuas de D_1 para D_2 e representado por $D_2^{D_1}$) tomando como portador o conjunto

$$\{f : D_1 \longrightarrow D_2 |\ f \text{ é uma função contínua}\}$$

e definindo

ordem: $f \sqsubseteq g \triangleq \langle \forall\, x \,:\, x \in D_1 :\ f\, x \sqsubseteq_2 g\, x \rangle$

mínimo: $\bot \triangleq \underline{\bot_2}$ (i.e., a função que a qualquer argumento faz corresponder $\bot_2$)

supremo: $\langle \bigsqcup\ n \,:\, n \geq 0 :\ f_n \rangle\, x \triangleq \langle \bigsqcup_2\ n \,:\, n \geq 0 :\ f_n\, x \rangle$

- Mostre que, de acordo com a definição dada acima, supremo de uma cadeia ascendente de funções contínuas é ainda uma função contínua.
  Prova:

$$\begin{array}{ll} & \langle \bigsqcup\ n \,:\, n \geq 0 :\ f_n \rangle\, \langle \bigsqcup\ m \,:\, m \geq 0 :\ x_m \rangle \\ = & \quad \{ \text{ definição de } \langle \bigsqcup\ n \,:\, n \geq 0 :\ f_n \rangle \ \} \\ & \langle \bigsqcup\ n \,:\, n \geq 0 :\ f_n\, \langle \bigsqcup\ m \,:\, m \geq 0 :\ x_m \rangle \rangle \\ = & \quad \{ \text{ cada } f_n \text{ é contínua } \} \\ & \langle \bigsqcup\ n \,:\, n \geq 0 :\ \langle \bigsqcup\ m \,:\, m \geq 0 :\ f_n\, x_m \rangle \rangle \\ = & \quad \{ \text{ troca } \} \\ & \langle \bigsqcup\ m \,:\, m \geq 0 :\ \langle \bigsqcup\ n \,:\, n \geq 0 :\ f_n\, x_m \rangle \rangle \\ = & \quad \{ \text{ definição de } \langle \bigsqcup\ n \,:\, n \geq 0 :\ f_n \rangle \ \} \\ & \langle \bigsqcup\ m \,:\, m \geq 0 :\ \langle \bigsqcup\ n \,:\, n \geq 0 :\ f_n \rangle\, x_m \rangle \end{array}$$

- Mostre que a função $\mathsf{ev} : E^D \times D \longrightarrow E$ definida por

$$\mathsf{ev}\,(f, x) \ =\ f\, x$$

é contínua.
Prova:

$$\begin{array}{ll} & \mathsf{ev}\, \langle \bigsqcup\ n \,:\, n \geq 0 :\ (f_n, x_n) \rangle \\ = & \quad \{ \text{ definição de supremo num domínio produto } \} \\ & \mathsf{ev}\, (\langle \bigsqcup\ k \,:\, k \geq 0 :\ f_k \rangle, \langle \bigsqcup\ m \,:\, m \geq 0 :\ x_m \rangle) \\ = & \quad \{ \text{ definição de ev } \} \\ & \langle \bigsqcup\ k \,:\, k \geq 0 :\ f_k \rangle\, \langle \bigsqcup\ m \,:\, m \geq 0 :\ x_m \rangle \\ = & \quad \{ \text{ definição de supremo num domínio exponencial } \} \\ & \langle \bigsqcup\ k \,:\, k \geq 0 :\ f_k\, \langle \bigsqcup\ m \,:\, m \geq 0 :\ x_m \rangle \rangle \end{array}$$

$$
\begin{array}{ll}
= & \{ \text{ cada } f_k \text{ é contínua } \} \\
& \langle \bigsqcup\ k \ :\ k \geq 0 :\ \langle \bigsqcup\ m \ :\ m \geq 0 :\ f_k\, x_m \rangle \rangle \\
= & \{ \text{ troca } \} \\
& \langle \bigsqcup\ n \ :\ n \geq 0 :\ f_n\, x_n \rangle \\
= & \{ \text{ definição de ev } \} \\
& \langle \bigsqcup\ n \ :\ n \geq 0 :\ \mathsf{ev}\,(f_n, x_n) \rangle
\end{array}
$$

- Considere a função curry : $E^{C \times D} \longrightarrow E^{D^C}$ definida por

$$\mathsf{curry}\, f\, c \ =\ \langle \lambda\, d \ :\ d \in D :\ f\,(c,d) \rangle$$

  Mostre que curry f é contínua.

Aula 13. [(TP) Seg, 26 Mai 2008, 9-11 h]

Sumário:
Conclusão da aula anterior: resolução de exercícios (construções sobre domínios)

Exercício (domínio soma).

Domínio soma
Dados dois domínios $(D_1, \sqsubseteq_1, \bot_1, \bigsqcup_1)$ e $(D_2, \sqsubseteq_2, \bot_2, \bigsqcup_2)$, define-se um novo domínio (dito a *soma separada* de D_1 e D_2) tomando como portador o conjunto

$$\{\iota_1\, x |\ x \in D_1\} \cup \{\iota_2\, x |\ x \in D_2\} \cup \bot$$

e definindo

ordem: $x \sqsubseteq y \ \triangleq\ x = \bot \vee \langle \exists\, i\ :\ i \in \{1,2\} :\ x = \iota_1\, x' \wedge y = \iota_1\, y' \wedge x' \sqsubseteq_i y' \rangle$

mínimo: $\bot$

supremo: ou é $\bot$ (se a cadeia consistir neste único elemento) ou é o supremo correspondente em D_1 ou D_2 (note-se que numa cadeia em $D_1 + D_2$ apenas pode conter $\bot$ e um conjunto de elementos ou de D_1 ou de D_2).

- Mostre que a definição acima define efectivamente um domínio.
- Mostre que as funções canónicas de injecção, ι_1 e ι_2, são contínuas, mas não estritas.
- Considere a função

$$[f, g]\, x \ =\ \begin{cases} \bot & \Leftarrow x = \bot \\ f\, x' & \Leftarrow x = \iota_{x'} \\ g\, y' & \Leftarrow x = \iota_{y'} \end{cases}$$

  Mostre que esta função é monótona e contínua em ambos os seus argumentos.

Nota:
Repare-se que a propriedade universal que, no universo dos conjuntos e das funções, associamos a esta construção não é válida, pelo que a soma separada não é um coproduto no universo dos domínios e das funções contínuas. Temos apenas,

$$k = [f, g] \ \Leftarrow\ k \cdot \iota_1 = f \wedge k \cdot \iota_2 = g$$

i.e., o lado esquerdo determina k completamente, dados f e g, mas o lado direito não determina $k\bot$. Reporte-se ao que estudou em Cálculo de Progamas e procure responder:

- Que leis, do *kit* associado à soma de conjuntos que estudou, permanecem válidas neste contexto?
- Mostre que a propriedade universal acima é válida se f e g forem funções estritas.
- Existe uma outra forma de definir a soma de dois domínios que consiste em identificar $\bot_1$ e $\bot_2$ num único elemento $\bot$. Mostre que esta forma de soma, dita *soma partilhada*, satisfaz a propriedade universal quando restrita ao universo dos domínios e das funções contínuas estritas (i.e., a *soma partilhada* é um *coproduto*) nesse universo.

Exercício (Combinador *ponto fixo*).
Dado um domínio D define-se a função

$$\mu : D^D \longrightarrow D$$

que a cada função contínua $f : D \longrightarrow D$ faz corresponder o seu menor ponto fixo, i.e., $\mu\, f = \mu_f$.

- Mostre que μ é monótona.
  Prova:

$$
\begin{array}{ll}
 & f_1 \sqsubseteq f_2 \\
\Rightarrow & \quad \{ \text{ definição de } \sqsubseteq \} \\
 & f_1\, \mu\, f_2 \sqsubseteq f_2 \mu\, f_2 \\
\Rightarrow & \quad \{ \ \mu\, f_2 \text{ é pre-ponto fixo de } f_2 \ \} \\
 & f_1\, \mu\, f_2 \sqsubseteq f_2 \mu\, f_2 \sqsubseteq \mu\, f_2 \\
\Rightarrow & \quad \{ \ \mu\, f_2 \text{ é também um pre-ponto fixo de } f_1 \ \} \\
 & \mu\, f_1 \sqsubseteq \mu\, f_2
\end{array}
$$

- Mostre que μ é contínua.
  Prova: Sendo μ monótona, $\langle \bigsqcup\ n\ :\ n \leq 0 :\ \mu_{f_n} \rangle\ \sqsubseteq\ \mu \langle \bigsqcup\ n\ :\ n \leq 0 :\ f_n \rangle$ (porquê?). Basta, pois, verificar a outra inequação:

$$\mu \langle \bigsqcup\ n\ :\ n \leq 0 :\ f_n \rangle\ \sqsubseteq\ \langle \bigsqcup\ n\ :\ n \leq 0 :\ \mu_{f_n} \rangle$$

  Para isso, verifiquemos que $\langle \bigsqcup\ n\ :\ n \leq 0 :\ \mu\, f_n \rangle$ é pre-ponto fixo da função $\langle \bigsqcup\ n\ :\ n \leq 0 :\ f_n \rangle$. Como $\mu \langle \bigsqcup\ n\ :\ n \leq 0 :\ f_n \rangle$ é o menor pre-ponto fixo dessa mesma função, concluimos.

$$
\begin{array}{ll}
 & \langle \bigsqcup\ m\ :\ m \leq 0 :\ f_m \rangle \langle \bigsqcup\ n\ :\ n \leq 0 :\ \mu\, f_n \rangle \\
= & \quad \{ \text{ definição de supremo num domínio exponencial } \} \\
 & \langle \bigsqcup\ m\ :\ m \leq 0 :\ f_m \langle \bigsqcup\ n\ :\ n \leq 0 :\ \mu\, f_n \rangle \rangle \\
= & \quad \{ \text{ cada } f_n \text{ é contínua } \} \\
 & \langle \bigsqcup\ m\ :\ m \leq 0 :\ \langle \bigsqcup\ n\ :\ n \leq 0 :\ f_m\, \mu\, f_n \rangle \rangle \\
= & \quad \{ \text{ troca } \} \\
 & \langle \bigsqcup\ k\ :\ k \leq 0 :\ f_k\, \mu\, f_k \rangle \\
\sqsubseteq & \quad \{ \text{ definição de pre-ponto fixo para cada } f_k \ \} \\
 & \langle \bigsqcup\ k\ :\ k \leq 0 :\ \mu\, f_k \rangle
\end{array}
$$

Aula 14. [(T) Seg, 26 Mai 2008, 11-13 h]

Sumário:
Domínios planos e operação de *lifting*. Uma meta-linguagem para a escrita de funções contínuas.

Domínios planos
Qualquer conjunto X pode ser transformado num domínio, dito *plano*, por inclusão de um elemento mínimo. Assim,

portador: $X_\perp \triangleq \{\lfloor x \rfloor | \, x \in X\} \cup \{\perp\}$

ordem: $d \sqsubseteq d' \triangleq d = d' \vee d = \perp$

onde $\lfloor \, \rfloor : X \longrightarrow X_\perp$ é injectiva.
Esta transformação, quando aplicada a um domínio, conhecida por *lifting*, dizendo-se $D_\perp$ o *lift* do domínio original D.

Exercício.
Considere a seguinte definição da função condicional $\mathsf{if\ \_\ then\ \_\ else\ \_} : \mathbb{B}_\perp \times (D \times D) \longrightarrow D$ sobre um domínio D:

$$\mathsf{if}\, b\, \mathsf{then}\, x\, \mathsf{else}\, y \triangleq \begin{cases} x & \Leftarrow b = \mathsf{true} \\ y & \Leftarrow b = \mathsf{false} \\ \perp & \Leftarrow b = \perp \end{cases}$$

Mostre que a função é contínua nos seus dois argumentos (do tipo $\mathbb{B}_\perp$ e $D \times D$, respectivamente).

Exercício.
Seja $f : D \longrightarrow E$ uma função contínua. Mostre que a função $f^\perp : D_\perp \longrightarrow E$ dada por

$$f^\perp x \triangleq \begin{cases} f\, d & \Leftarrow x = \lfloor d \rfloor \text{ para algum } d \in D \\ \perp \Leftarrow \text{ caso contrário} & \end{cases}$$

é ainda contínua.

Nota
O *lifting* de funções, via $f^\perp$ acima, é extremamente útil quando se pretende raciocinar em ou sobre domínios. Por exemplo, a interpretação da notação $let \cdots in \cdots$ será

$$let\,(x = y)\,in\,e \triangleq \langle \lambda\, x \,::\, e \rangle^\perp\, y$$

o que garante que apenas no caso de y não ser indefinido (i.e., $y \neq \perp$) será usado na determinação do valor da expressão e.
Na prática esta construção é fundamental para estender a $D_\perp$ operações em D. Por exemplo, a extensão $\vee_\perp : \mathbb{B}_\perp \longrightarrow \mathbb{B}_\perp$ da usual disjunção booleana será

$$x\_1 \vee_\perp x_2 \triangleq let\,(t_1 = x_1, t_2 = x_2)\,in\,\lfloor t_1 \vee t_2 \rfloor$$

Especificação de funções contínuas
O requisito de continuidade das funções usadas em Semântica é essencial (para se poder determinar o siginificado de definições recursivas como pontos fixos). No entanto, a repetição exaustiva de provas de continuidade é morosa e não agiliza o raciocínio em Semântica. Esta situação pode ser contornada, contudo, se na especificação das funções usarmos apenas um conjunto de construções sintáticas que se prova serem invariantes para a continuidade. I.e., quando aplicadas a funções contínuas geram ainda funções contínuas.

Por exemplo, a prova de que a composição de funções contínuas é ainda contínua pode ser substituída pela observação de que

$$f \cdot g \;=\; \langle \lambda\, x \;::\; f\, g\, x \rangle$$

e pelo facto de o processo de construção de abstrações-λ garantir, como se mostra abaixo, a preservação da continuidade.

São, pois, as seguintes as construções admissíveis para especificar funções contínuas:

Variáveis A expressão x, onde x denota uma variável num domínio D, é admissível uma vez que a abstração $\langle \lambda\, x \;:\; x \in D :\; y \rangle$ é ou a função identidade ou uma função constante.

Constantes I.e., expressões que denotam elementos fixos de um domínio: por exemplo 5, $\bot$, true, mas também eval, π_1, ... que também denotam elementos fixos de um domínio (o domínio exponencial).

Tuplos I.e., expressões $(e_1, \cdots, e_n)$ que são valores no domínio produto $D_1 \times \cdots D_n$. Estas expressões são contínuas numa variável $x \in D$ porque

$$\begin{array}{ll}
 & \langle \lambda\, x \;:\; x \in D :\; (e_1, \cdots, e_n) \rangle \quad \text{é contínuo} \\
\Leftrightarrow & \quad \{ \ldots \} \\
 & \langle \forall\, i \;:\; 1 \le i \le n :\; \pi_i\, \langle \lambda\, x \;:\; x \in D :\; (e_1, \cdots, e_n) \rangle \quad \text{é contínuo} \rangle \\
\Leftrightarrow & \quad \{ \ldots \} \\
 & \langle \forall\, i \;:\; 1 \le i \le n :\; \langle \lambda\, x \;:\; x \in D :\; e_i \rangle \quad \text{é contínuo} \rangle \\
\Leftrightarrow & \quad \{ \ldots \} \\
 & \langle \forall\, i \;:\; 1 \le i \le n :\; e_i \quad \text{é contínuo} \rangle \\
\Leftrightarrow & \quad \{ \text{por hipótese cada } e_i \text{ é contínua em } x \} \\
 & \mathsf{true}
\end{array}$$

Aplicação Dada uma função contínua h, $h\, e$ é contínuo em x porque

$$\begin{array}{ll}
 & h\, e \quad \text{é contínuo} \\
\Leftrightarrow & \quad \{ \ldots \} \\
 & h \cdot \langle \lambda\, x \;:\; x \in D :\; e \rangle \quad \text{é contínuo} \\
\Leftarrow & \quad \{ \ldots \} \\
 & \langle \lambda\, x \;:\; x \in D :\; e \rangle \quad \text{é contínuo} \\
\Leftrightarrow & \quad \{ \ldots \} \\
 & e \quad \text{é contínuo em } x \\
\Leftrightarrow & \quad \{ \text{por hipótese} \} \\
 & \mathsf{true}
\end{array}$$

Especificação de funções contínuas (cont.)

Abstração-λ Se a expressão $e \in E$ é contínua nas suas variáveis, então claramente $\langle \lambda\ y\ :\ y \in D :\ e \rangle$ também o será. Além disso o próprio processo de construção de abstracções-λ é contínuo na sua variável x. Se x coincide com a variável y o resultado é imediato. Caso contrário,

$$
\begin{array}{ll}
 & \langle \lambda\ y\ ::\ e \rangle \quad \text{é contínua em } x \\
\Leftrightarrow & \quad \{ \ldots \} \\
 & \langle \lambda\ x\ ::\ \langle \lambda\ y\ ::\ e \rangle \rangle \quad \text{é contínuo} \\
\Leftrightarrow & \quad \{ \ldots \} \\
 & \mathsf{curry}\ \langle \lambda\ x, y\ ::\ e \rangle \quad \text{é contínuo} \\
\Leftarrow & \quad \{ \ldots \} \\
 & \langle \lambda\ x, y\ ::\ e \rangle \quad \text{é contínuo} \\
\Leftrightarrow & \quad \{ \ldots \} \\
 & e \quad \text{é contínua em } x \text{ e } y \\
\Leftrightarrow & \quad \{ \text{por hipótese} \} \\
 & \mathsf{true}
\end{array}
$$

Exercício. Preencha todas as reticências nos argumentos acima, justificando os passos dados.

Exercício. Mostre que se se definir uma função entre domínios na metalinguagem descrita acima mas enriquecida com declarações locais da forma $let\ (x = d)\ in\ e$, essa função ainda será contínua.

Aula 15. [(T) Ter, 27 Mai 2008, 9-11 h]

Sumário:
Equivalência entre as semântica operacional e denotacional da linguagem LIS.

Motivação.
A definição da semântica *denotacional* para o ciclo while partiu da seguinte observação intuída a partir da sua semântica *operacional*:

$$[\![\mathsf{while}\ B\ \mathsf{do}\ C]\!] \ = \ [\![\mathsf{if}\ B\ \mathsf{then}\ C; \mathsf{while}\ B\ \mathsf{do}\ C\ \mathsf{else}\ \mathsf{skip}]\!] \tag{11}$$

Mais tarde demos uma definição *não recursiva* para a semântica *denotacional* do ciclo, i.e.,

$$[\![\mathsf{while}\ B\ \mathsf{do}\ C]\!] \ = \ \mu_f$$

onde

$$f\,\alpha \ = \ \langle \lambda\ s \ : \ s \in \Sigma : \begin{cases} \alpha \bullet [\![C]\!]\, s & \Leftarrow [\![B]\!]\, s \\ s & \Leftarrow \neg [\![B]\!]\, s \end{cases} \rangle$$

com base na qual podemos provar a equação (11):

$$\begin{array}{ll} & [\![\mathsf{if}\ B\ \mathsf{then}\ C; \mathsf{while}\ B\ \mathsf{do}\ C\ \mathsf{else}\ \mathsf{skip}]\!] \\ \Leftrightarrow & \quad \{\ \text{semântica denotacional do condicional, composição sequencial e skip}\ \} \\ & \langle \lambda\ s \ : \ s \in \Sigma : \begin{cases} [\![\mathsf{while}\ B\ \mathsf{do}\ C]\!] \bullet [\![C]\!]\, s & \Leftarrow [\![B]\!]\, s \\ s & \Leftarrow \neg [\![B]\!]\, s \end{cases} \rangle \\ \Leftrightarrow & \quad \{\ \text{definição de } f\ \} \\ & f\, [\![\mathsf{while}\ B\ \mathsf{do}\ C]\!] \\ \Leftrightarrow & \quad \{\ [\![\mathsf{while}\ B\ \mathsf{do}\ C]\!] \text{ é ponto fixo de } f\ \} \\ & [\![\mathsf{while}\ B\ \mathsf{do}\ C]\!] \end{array}$$

Resultados.

Equivalência semântica para a linguagem LIS (expressões)
$$[\![E]\!]\, s = v \ \Leftrightarrow \ \langle E, s\rangle \Downarrow \langle v, s\rangle$$

Equivalência semântica para a linguagem LIS (comandos)
(Correcção) $[\![C]\!]\, s = s' \ \Rightarrow \ \langle C, s\rangle \Downarrow \langle V, s'\rangle$ (prova por *indução estrutural* sobre os comandos)
(Adequação) $\langle C, s\rangle \Downarrow \langle V, s'\rangle \ \Rightarrow \ [\![C]\!]\, s = s'$ (prova por *indução sobre as regras* que definem a relação $\Downarrow$)

Aula 16. [(T) Qua, 28 Mai 2008, 11-13 h]

Sumário:
Introdução à semântica das linguagens funcionais. Semânticas operacional e denotacional da avaliação de funções *por passagem de valores* (*call-by-value*).

LFS **: Sintaxe**

$$T \quad ::= \quad x \mid n \mid b \mid T\ iop\ T \mid T\ brel\ T \mid \mathsf{if}\ B\ \mathsf{then}\ T\ \mathsf{else}\ T \mid f(T, T, \cdots, T)$$

com $n \in \mathbb{Z}$, $b \in \mathbb{B}$, x pertence a um conjunto $Var = \{x_1, x_2, \cdots\}$ de variáveis e f a um conjunto $FVar = \{f_1, f_2, \cdots\}$ de identificadores de funções. Representa-se por $ar(f)$ a aridade da função f, i.e., o seu número de argumentos.

LFS **(termos fechados): Semântica operacional** A avaliação de termos LFS é sempre relativa a um determinado conjunto de declarações de funções:

$$\begin{aligned} f_1(x_1, \cdots, x_{ar(f_1)}) &= e_1 \\ \cdots &= \cdots \\ f_n(x_1, \cdots, x_{ar(f_n)}) &= e_n \end{aligned}$$

$$(EVcon) \qquad c \Downarrow c \quad (c \in \mathbb{Z} \cup \mathbb{B})$$

$$(EVop) \qquad \frac{t_1 \Downarrow v_1 \quad t_2 \Downarrow v_2}{t_1\ op\ t_2 \Downarrow v} \quad \Leftarrow v = v_1\ op\ v_2$$

$$(EVif1) \qquad \frac{tb \Downarrow \mathsf{true} \quad t_1 \Downarrow v_1}{\mathsf{if}\ tb\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 \Downarrow v_1}$$

$$(EVif2) \qquad \frac{tb \Downarrow \mathsf{false} \quad t_2 \Downarrow v_2}{\mathsf{if}\ tb\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 \Downarrow v_2}$$

$$(EVfun) \qquad \frac{t_1 \Downarrow v_1 \quad t_2 \Downarrow v_2 \quad \cdots \quad t_{ar(f)} \Downarrow v_{ar(f)} \quad e[v_1/x_1, v_2/x_2, \cdots, v_{ar(f)}/x_{ar(f)}] \Downarrow v}{f(t_1, t_2, \cdots t_{ar(f)}) \Downarrow v}$$

Exercício.
Mostre que a semântica proposta para LFS é determinística.

LFS **: Semântica denotacional**
O significado dos termos é relativo a um ambiente que associa as variáveis em Var às suas instâncias

$$\rho : Var \longrightarrow V$$

onde $V = \mathbb{Z} \cup \mathbb{B}$, e os idenficadores de função, em $FVar$, à semântica das respectivas declarações, i.e., para um programa contendo declarações das funções f_1 a f_k, um tuplo

$$\phi = \langle \phi_1, \phi_2, \cdots, \phi_k \rangle \quad \text{onde} \quad \phi_i : V^{ar(f_i)} \longrightarrow V_\perp$$

O significado de um termo t será, assim, uma função

$$[\![t]\!] : Fenv \longrightarrow Venv \longrightarrow V_\perp$$

onde $Venv$ e $Fenv$ designam, respectivamente, o tipo dos ambientes associados a variáveis e a funções que formam o contexto do termo. A função semântica é, assim, definida por

$$\begin{aligned}
[\![v]\!]\,\phi\,\rho &= \lfloor v \rfloor \\
[\![x]\!]\,\phi\,\rho &= \lfloor \rho\, x \rfloor \\
[\![t_1 op\, t_2]\!]\,\phi\,\rho &= [\![t_1]\!]\phi\rho\ op_\perp\ [\![t_2]\!]\phi\rho \\
[\![\mathsf{if}\ tb\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2]\!]\,\phi\,\rho &= \begin{cases} [\![tb]\!]\phi\rho & \Rightarrow\ [\![t_1]\!]\phi\rho \\ \neg[\![tb]\!]\phi\rho & \Rightarrow\ [\![t_2]\!]\phi\rho \end{cases} \\
[\![f(t_1, \cdots, t_{ar(f)})]\!]\,\phi\,\rho &= let\ (v_1 = [\![t_1]\!]\phi\rho, \cdots\ v_{ar(f)} = [\![t_{ar(f)}]\!]\phi\rho)\ in\ \ \phi_i(v_1, \cdots, v_{ar(f)})
\end{aligned}$$

Note-se que um termo em LFS é avaliado sempre em referência ao conjunto de funções declaradas:

$$\begin{aligned}
f_1(x_1, \cdots, x_{ar(f_1)}) &= e_1 \\
\cdots &= \cdots \\
f_k(x_1, \cdots, x_{ar(f_k)}) &= e_k
\end{aligned}$$

Estas declarações podem ser vistas como um sistema de equações recursivas nas variáveis f_1 a f_k a satisfazer por um ambiente $\phi = \langle \phi_1, \phi_2, \cdots, \phi_k \rangle$:

$$\begin{aligned}
\langle \forall\, v_i\ :\ v_i \in V :\ \phi_1(v_1, \cdots, v_{ar(f_1)}) &= [\![e_1]\!]\phi\rho[v_1/x_1, \cdots, v_{ar(f_1)}/x_{ar(f_1)}]\rangle \\
\cdots &= \cdots \\
\langle \forall\, v_i\ :\ v_i \in V :\ \phi_k(v_1, \cdots, v_{ar(f_k)}) &= [\![e_1]\!]\phi\rho[v_1/x_1, \cdots, v_{ar(f_k)}/x_{ar(f_k)}]\rangle
\end{aligned}$$

Apesar deste sistema poder admitir mais que uma solução, podemos calcular uma solução canónica como o menor ponto fixo da função

$$H : Fenv \longrightarrow Fenv$$

definida por

$$\begin{aligned}
H\,\phi\ = \langle\ & \\
& \langle \lambda\, v_1, \cdots, v_{ar(f_1)}) \ ::\ [\![e_1]\!]\phi\rho[v_1/x_1, \cdots, v_{ar(f_1)}/x_{ar(f_1)}]\rangle, \\
& \cdots \\
& \langle \lambda\, v_1, \cdots, v_{ar(f_k)}) \ ::\ [\![e_k]\!]\phi\rho[v_1/x_1, \cdots, v_{ar(f_k)}/x_{ar(f_k)}]\rangle \\
\rangle &
\end{aligned}$$

Assim, o ambiente de avaliação determinado por um conjunto de declarações (i.e., pelo programa funcional) será

$$\phi \;=\; \mu_H$$

Note-se, por fim, que para um termo fechado t, $[\![t]\!]\phi\rho$ é sempre um elemento de $V_\perp$, calculado em função do ambiente ϕ, mas independente de ρ.

Exemplo.

$$\begin{aligned} f\,x &\triangleq f\,x + 3 \\ g\,x &\triangleq 5 \end{aligned}$$

A semântica desta declaração é um ambiente

$$\delta = (\delta_1, \delta_2) \in V_\perp^V \times V_\perp^V$$

onde (δ_1, δ_2) é o menor ponto fixo da função

$$\begin{array}{rcl} H\,\phi & = & (\langle \lambda\, m \;:\; m \in \mathbb{Z} :\; [\![f\,x + 3]\!]\phi\rho[m/x]\rangle, \langle \lambda\, m \;:\; m \in \mathbb{Z} :\; [\![5]\!]\phi\rho[m/x]\rangle) \\ & = & (\langle \lambda\, m \;:\; m \in \mathbb{Z} :\; [\![f\,x]\!]\phi\rho[m/x] +_\perp \lfloor 3 \rfloor\rangle, \langle \lambda\, m \;:\; m \in \mathbb{Z} :\; \lfloor 5 \rfloor\rangle) \\ & = & (\langle \lambda\, m \;:\; m \in \mathbb{Z} :\; \phi_1\, m +_\perp \lfloor 3 \rfloor\rangle, \langle \lambda\, m \;:\; m \in \mathbb{Z} :\; \lfloor 5 \rfloor\rangle) \end{array}$$

Isto é,

$$(\delta_1, \delta_2) \;=\; \mu_H \;=\; (\perp, \langle \lambda\, m \;:\; m \in \mathbb{Z} :\; \lfloor 5 \rfloor\rangle)$$

Então

$$\begin{array}{rcl} [\![g\,f\,10]\!]\delta\rho & = & let\,(z = [\![f\,10]\!]\delta\rho)\;\; in\;\; \delta_2\, z \\ & = & \delta_2\, \perp \\ & = & \perp \end{array}$$

Exercício.
Justifique o cálculo do menor ponto fixo de H no exemplo acima.

Exercício.
Mostre que a denotação $[\![t]\!]$ de um termo t arbitrário é uma função contínua no domínio $V_\perp^{Venv^{Fenv}}$.

Exercício.
Mostre que a função H definida acima é uma função contínua.

Aula 17. [(T) Seg, 2 Jun 2008, 9-11 h]

Sumário:
Não houve aula: participação do docente na reunião da Assembleia da Universidade.

Aula 18. [(TP) Seg, 2 Jun 2008, 11-13 h]

Sumário:
Não houve aula: participação do docente na reunião da Assembleia da Universidade.

Aula 19. [(TP) Ter, 3 Jun 2008, 9-11 h]

Sumário:
Resolução de exercícios.

Aula 20. [(TP) Qua, 4 Jun 2008, 11-13 h]

Sumário:
Estudo de caso: semânticas operacional e denotacional da avaliação de funções *por passagem de referências* (*call-by-name*).

LFS **(termos fechados): Semântica operacional** A avaliação de termos LFS é sempre relativa a um determinado conjunto de declarações de funções:

$$\begin{aligned} f_1(x_1, \cdots, x_{ar(f_1)}) &= e_1 \\ \cdots &= \cdots \\ f_n(x_1, \cdots, x_{ar(f_n)}) &= e_n \end{aligned}$$

$$(EVcon) \qquad c \Downarrow c \quad (c \in \mathbb{Z} \cup \mathbb{B})$$

$$(EVop) \qquad \frac{t_1 \Downarrow v_1 \quad t_2 \Downarrow v_2}{t_1 \, op \, t_2 \Downarrow v} \Leftarrow v = v_1 \, op \, v_2$$

$$(EVif1) \qquad \frac{tb \Downarrow \mathsf{true} \quad t_1 \Downarrow v_1}{\mathsf{if}\ tb\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 \Downarrow v_1}$$

$$(EVif2) \qquad \frac{tb \Downarrow \mathsf{false} \quad t_2 \Downarrow v_2}{\mathsf{if}\ tb\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 \Downarrow v_2}$$

$$(EVfun) \qquad \frac{e[t_1/x_1, t_2/x_2, \cdots, t_{ar(f)}/x_{ar(f)}] \Downarrow v}{f(t_1, t_2, \cdots t_{ar(f)}) \Downarrow v}$$

Exercício.
Mostre que a semântica operacional proposta para LFS com *call-by-name* é determinística.

LFS **: Semântica denotacional**
Tal como no caso anterior, o significado dos termos é relativo a um ambiente que associa as variáveis em Var às suas instâncias (mas agora definido de modo ligeiramente diferente; porquê?)

$$\rho : Var \longrightarrow V_\perp$$

onde $V = \mathbb{Z} \cup \mathbb{B}$, e os idenficadores de função, em $FVar$, à semântica das respectivas declarações, i.e., para um programa contendo declarações das funções f_1 a f_k, um tuplo

$$\phi = \langle \phi_1, \phi_2, \cdots, \phi_k \rangle \quad \text{onde} \quad \phi_i : V_\perp^{ar(f_i)} \longrightarrow V_\perp$$

O significado de um termo t será, assim, uma função

$$[\![t]\!] : Fenv \longrightarrow Venv \longrightarrow V_\perp$$

onde $Venv$ e $Fenv$ designam, respectivamente, o tipo dos ambientes associados a variáveis e a funções que formam o contexto do termo. A função semântica é, assim, definida por

$$\begin{aligned}
[\![v]\!]\,\phi\,\rho &= \lfloor v \rfloor \\
[\![x]\!]\,\phi\,\rho &= \lfloor \rho\, x \rfloor \\
[\![t_1 op\, t_2]\!]\,\phi\,\rho &= [\![t_1]\!]\phi\rho\ op_\perp\ [\![t_2]\!]\phi\rho \\
[\![\mathsf{if}\ tb\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2]\!]\,\phi\,\rho &= \begin{cases} [\![tb]\!]\phi\rho & \Rightarrow\ [\![t_1]\!]\phi\rho \\ \neg[\![tb]\!]\phi\rho & \Rightarrow\ [\![t_2]\!]\phi\rho \end{cases} \\
[\![f(t_1, \cdots, t_{ar(f)})]\!]\,\phi\,\rho &= \phi_i([\![t_1]\!]\,\phi\,\rho, \cdots, [\![t_{ar(f)}]\!]\,\phi\,\rho)
\end{aligned}$$

Exemplo.

$$\begin{aligned}
f\,x &\triangleq f\,x + 3 \\
g\,x &\triangleq 5
\end{aligned}$$

A semântica desta declaração é um ambiente

$$\delta = (\delta_1, \delta_2) \in V_\perp^V \times V_\perp^V$$

onde (δ_1, δ_2) é o menor ponto fixo da função

$$\begin{aligned}
H\,\phi &= (\langle \lambda\, m\ :\ m \in \mathbb{Z}_\perp :\ [\![f\,x + 3]\!]\phi\rho[m/x] \rangle, \langle \lambda\, m\ :\ m \in \mathbb{Z}_\perp :\ [\![5]\!]\phi\rho[m/x] \rangle) \\
&= (\langle \lambda\, m\ :\ m \in \mathbb{Z}_\perp :\ [\![f\,x]\!]\phi\rho[m/x] +_\perp \lfloor 3 \rfloor \rangle, \langle \lambda\, m\ :\ m \in \mathbb{Z}_\perp :\ \lfloor 5 \rfloor \rangle) \\
&= (\langle \lambda\, m\ :\ m \in \mathbb{Z}_\perp :\ \phi_1\, m +_\perp \lfloor 3 \rfloor \rangle, \langle \lambda\, m\ :\ m \in \mathbb{Z}_\perp :\ \lfloor 5 \rfloor \rangle)
\end{aligned}$$

Isto é,

$$(\delta_1, \delta_2) \ =\ \mu_H \ =\ (\perp, \langle \lambda\, m\ :\ m \in \mathbb{Z}_\perp :\ \lfloor 5 \rfloor \rangle)$$

Então

$$\begin{aligned}
&[\![g\ f\ 10]\!]\delta\rho \\
&\quad = \quad \delta_2\,\delta_1 \\
&\quad = \quad \delta_2\,\perp \\
&\quad = \quad \lfloor 5 \rfloor
\end{aligned}$$

Exercício.
Calcule $f\ 4$ assumindo a declaração

$$f\ x\ =\ \mathsf{if}\ x = 0\ \mathsf{then}\ 1\ \ \mathsf{else}\ x * f(x-1)$$

Sugestão.
Considere $\mathbb{Z}_\perp^\mathbb{Z}$ como o tipo de $Fenv$ e calcule ϕ como o menor ponto fixo da função

$$\begin{array}{rcl}
H\,\phi & = & \langle \lambda\, m\ :\ m \in \mathbb{Z}_\perp :\ [\![\mathsf{if}\ x = 0\ \mathsf{then}\ 1\ \ \mathsf{else}\ x * f(x-1)]\!]\phi\rho[m/x]\rangle \\
& = & \langle \lambda\, m\ :\ m \in \mathbb{Z}_\perp :\ \begin{cases} \lfloor 1 \rfloor & \Leftarrow\ [\![x]\!]\phi\rho[m/x] = 0 \\ [\![x]\!]\phi\rho[m/x] \times_\perp \phi([\![x]\!]\phi\rho[m/x] - 1) & \Leftarrow\ [\![x]\!]\phi\rho[m/x] \neq 0 \end{cases}\rangle \\
& = & \langle \lambda\, m\ :\ m \in \mathbb{Z}_\perp :\ \begin{cases} \lfloor 1 \rfloor & \Leftarrow\ m = 0 \\ \lfloor m \rfloor \times_\perp \phi(m-1) & \Leftarrow\ m \neq 0 \end{cases}\rangle
\end{array}$$

Aula 21. [(T) Seg, 9 Jun 2008, 9 - 11 h]

Sumário:
Semântica das linguagens funcionais com tipos de ordem superior sob avaliação *eager*.

LFS++ **: Sintaxe**

$$\begin{array}{rcl} T & ::= & x \mid n \mid b \mid T\ iop\ T \mid T\ brel\ T \mid \mathsf{if}\ B\ \mathsf{then}\ T\ \mathsf{else}\ T \mid f(T,T,\cdots,T) \\ & & \mid (T,T) \mid \mathsf{fst} \mid \mathsf{snd} \\ & & \mid (T\ T) \mid \lambda x.T \mid \mathsf{let}\ (x \leftarrow T)\ \mathsf{in}\ T \mid \mathsf{rec}\ (y = \lambda x.T) \end{array}$$

com $n \in \mathbb{Z}$, $b \in \mathbb{B}$, x pertence a um conjunto $Var = \{x_1, x_2, \cdots\}$ de variáveis e f a um conjunto $FVar = \{f_1, f_2, \cdots\}$ de identificadores de funções.

Notas

- Termos representam não apenas valores primitivos (eg, inteiros e booleanos), mas também pares e funções.

  Formas Canónicas
  O conjunto das formas canónicas é o sub-conjunto dos termos fechados para os quais se considera desnecessário prosseguir a avaliação. Assim,

  - todos os valores booleanos ou inteiros são formas canónicas;
  - um tuplo de formas canónicas é uma forma canónica;
  - todas as expressões-λ fechadas são formas canónicas;

- A linguagem necessita, pois, de suporte sintático para exprimir *num termo* a entidade *declaração de função*. Tal é o propósito dos termos $\mathsf{rec}\ (y = \lambda x.T)$. Por exemplo a declaração

  $$f\,x \ = \ \mathsf{if}\ (x = 0)\ \mathsf{then}\ 1\ \mathsf{else}\ x * f(x-1)$$

  será representada pelo termo $\mathsf{rec}\ (f = \lambda x.\mathsf{if}\ (x = 0)\ \mathsf{then}\ 1\ \mathsf{else}\ x * f(x-1))$. Note-se que a sintaxe rec visa apenas sublinhar que a declaração abstraída neste termo pode ser recursiva na variável y. Uma alternativa a que se recorre por vezes é $\mathsf{fix}\ (y = \lambda x.T)$ para explicitar que o termo em causa representa um ponto fixo de uma equação em y.

- Para diferenciar as diferentes espécies de valores que podem resultar da avaliação de termos, é necessário introduzir um *sistema de tipos*:

  $$\tau \ ::= \ \mathsf{int} \mid \mathsf{bool} \mid \tau * \tau \mid \tau \longrightarrow \tau$$

- A introdução de tipos visa, precisamente, oferecer um critério de classificação dos termos e evitar expressões que não fazem sentido (e.g., somar um inteiro a uma função). Tal é concretizado via uma relação de tipagem $t : \tau$ (significando que o termo t habita o tipo τ):

LFS++ **: Regras de tipagem**

variáveis $x : \tau \quad \Leftarrow \mathsf{type}\, x = \tau$

valores e operações $n : \mathsf{int} \quad b : \mathsf{bool}$

$$\frac{t_0 : \mathsf{int} \quad t_1 : \mathsf{int}}{t_0\ iop\ t_1\ : \mathsf{int}} \qquad \frac{t_0 : \mathsf{int} \quad t_1 : \mathsf{int}}{t_0\ brel\ t_1\ : \mathsf{bool}}$$

$$\frac{t_0 : \mathsf{bool} \quad t_1 : \tau \quad t_2 : \tau}{\mathsf{if}\ t_0\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2\ :\ \tau}$$

tipo produto

$$\frac{t_1 : \tau_1 \quad t_2 : \tau_2}{(t_1, t_1) : \tau_1 * \tau_2} \qquad \frac{t : \tau_1 * \tau_2}{\mathsf{fst}\, t : \tau_1} \qquad \frac{t : \tau_1 * \tau_2}{\mathsf{snd}\, t : \tau_2}$$

tipo função

$$\frac{x : \tau_1 \quad t : \tau_2}{\lambda\, x.t : \tau_1 \longrightarrow \tau_2} \qquad \frac{t_1 : \tau_1 \longrightarrow \tau_2 \quad t_2 : \tau_2}{(t_1\, t_2) : \tau_2}$$

let

$$\frac{x : \tau_1 \quad t_1 : \tau_1 \quad t_2 : \tau_2}{\mathsf{let}\ (x \leftarrow t_1)\ \mathsf{in}\ t_2 : \tau_2}$$

rec

$$\frac{y : \tau \quad \lambda\, x.t : \tau}{\mathsf{rec}\, (y = \lambda\, x.t) : \tau}$$

LFS++ **(com avaliação *eager*): Semântica operacional**

(Efc) $\quad c \Downarrow c \quad (c$ forma canónica)

(Eop) $\quad \dfrac{t_1 \Downarrow v_1 \quad t_2 \Downarrow v_2}{t_1\, op\, t_2 \Downarrow v} \quad \Leftarrow v = v_1\, op\, v_2$

$(Eif1)$ $\quad \dfrac{tb \Downarrow \mathsf{true} \quad t_1 \Downarrow c_1}{\mathsf{if}\ tb\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 \Downarrow c_1}$

$(Eif2)$ $\quad \dfrac{tb \Downarrow \mathsf{false} \quad t_2 \Downarrow c_2}{\mathsf{if}\ tb\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 \Downarrow c_2}$

(Epr) $\quad \dfrac{t_1 \Downarrow c_1 \quad t_2 \Downarrow c_2}{(t_1, t_2) \Downarrow (c_1, c_2)}$

$(Ep1)$ $\quad \dfrac{t \Downarrow (c_1, c_2)}{\mathsf{fst}\, t \Downarrow c_1}$

$(Ep2)$ $\quad \dfrac{t \Downarrow (c_1, c_2)}{\mathsf{snd}\, t \Downarrow c_2}$

(Efu) $\quad \dfrac{t_1 \Downarrow \lambda\, x.t_1' \quad t_2 \Downarrow c_2 \quad t_1'[c_2/x] \Downarrow c}{(t_1\, t_2) \Downarrow c}$

(Elt) $\quad \dfrac{t_1 \Downarrow c_1 \quad t_2[c_1/x] \Downarrow c_2}{\mathsf{let}\ (x \leftarrow t_1)\ \mathsf{in}\ t_2 \Downarrow c_2}$

(Erc) $\quad \mathsf{rec}\, (y = \lambda\, x.t) \Downarrow \lambda\, x.(t[\mathsf{rec}\, (y = \lambda\, x.t)/y])$

LFS++ **(tipos): Semântica denotacional**
O universo de interpretação de cada tipo τ é um domínio de valores de τ contendo as denotações de todas as formas canónicas: a intuição é que um termo fechado ou é redutível a uma forma canónica ou a sua avaliação diverge (facto que, como habitualmente, se representa pelo mínimo $\perp$). Então,

$$\begin{aligned} V_{int} &= \mathbb{Z} \\ V_{bool} &= \mathbb{B} \\ V_{\tau_1 * \tau_2} &= V_{\tau_1} \times V_{\tau_2} \\ V_{\tau_1 \longrightarrow \tau_2} &= (V_{\tau_2})_{\perp}^{V_{\tau_1}} \end{aligned}$$

LFS++ **(termos): Semântica denotacional**

- Como usualmente os tipos contém variáveis livres, o seu significado é relativo a um ambiente de variáveis devidamente tipadas

$$Venv \;=\; \rho : Var \longrightarrow \langle \bigcup V_\tau \;::\; \tau \text{ é tipo} \rangle$$

- A cada termo tipado $t : \tau$ a semântica denotacional associa um elemento $[\![t]\!]\rho$ do domínio $(V_\tau)_\perp$ calculado relativamente ao ambiente de variáveis ρ.

A função semântica aplicada a um termo t do tipo τ

$$[\![t]\!] : Venv \longrightarrow (V_\tau)_\perp$$

é, assim, definida por

$$\begin{aligned}
[\![c]\!]\,\rho &= \lfloor v \rfloor \\
[\![x]\!]\,\rho &= \lfloor \rho\, x \rfloor \\
[\![t_1 op\, t_2]\!]\,\rho &= [\![t_1]\!]\rho \; op_\perp \; [\![t_2]\!]\rho \\
[\![\mathsf{if}\; tb \;\mathsf{then}\; t_1 \;\mathsf{else}\; t_2]\!]\,\rho &= \begin{cases} [\![tb]\!]\rho & \Rightarrow\; [\![t_1]\!]\rho \\ \neg [\![tb]\!]\rho & \Rightarrow\; [\![t_2]\!]\rho \end{cases} \\
[\![(t_1, t_2)]\!]\,\rho &= let\, v_1 = [\![t_1]\!]\rho, v_2 = [\![t_2]\!]\rho\, in\, \lfloor (v_1, v_2) \rfloor \\
[\![\mathsf{fst}(t)]\!]\,\rho &= \lfloor \pi_1\, [\![t]\!]\rho \rfloor \\
[\![\mathsf{snd}(t)]\!]\,\rho &= \lfloor \pi_2\, [\![t]\!]\rho \rfloor \\
[\![\lambda x.t]\!]\,\rho &= \lfloor \lambda\, v.[\![t]\!]\rho[v/x] \rfloor \\
[\![(t_1\, t_2)]\!]\,\rho &= let\, \phi = [\![t_1]\!]\rho, v = [\![t_2]\!]\, in\, \phi\, v \\
[\![\mathsf{let}\; (x \leftarrow t_1)\; \mathsf{in}\; t_2]\!]\,\rho &= let\, v = [\![t_1]\!]\rho\, in\, [\![t_2]\!]\rho[v/x] \\
[\![\lambda x.t]\!]\,\rho &= \lfloor \lambda\, v.[\![t]\!]\rho[v/x] \rfloor \\
[\![\mathsf{rec}\,(y = \lambda\, x.t)]\!]\,\rho &= \lfloor \langle \mu\; \phi \;::\; \lambda v\, .[\![t]\!]\rho\,[v/x, \phi/y] \rangle \rfloor
\end{aligned}$$

Aula 22. [(TP) Seg, 9 Jun 2008, 11 - 13 h]

Sumário:
Resultados de adequação.
Estudo de caso: semântica das linguagens funcionais com tipos de ordem superior sob avaliação *lazy*.

Adequação
A semântica denotacional diz-se *adequada* relativamente à operacional se

- para todo o termo t e forma canónica c,

$$t \Downarrow c \;\Rightarrow\; [\![t]\!]\rho = [\![c]\!]\rho \tag{12}$$

- para todo o termo $t : \tau$,

$$t \Downarrow \;\Leftrightarrow t \Downarrow^{Den} \tag{13}$$

onde as relações de *convergência* se definem por

$$\begin{array}{rcl} t \Downarrow & \triangleq & \langle \exists\, c \;:\; c \text{ é forma canónica} :\; t \Downarrow c \rangle \\ t \Downarrow^{Den} & \triangleq & \langle \exists\, v \;:\; v \in V_\tau :\; [\![t]\!]\rho = \lfloor v \rfloor \rangle \end{array}$$

Exercício. Porque razão não faz sentido substituir a implicação por uma equivalência na equação (12)?

Exercício. Mostre que para todo o termo t e $n \in \mathbb{Z}$ se tem

$$t \Downarrow n \;\Leftrightarrow\; [\![t]\!]\rho = [\![n]\!]\rho$$

Exercício. Esboce a demonstração do resultado de adequação acima enunciado (caso geral).

Sintaxe
A única diferença sintática relativamente ao caso anterior é a possibilidade de se permitir uma versão mais *liberal* da definição recursiva

$$\mathsf{rec}\,(y = t)$$

onde t já não designa necessariamente uma função (i.e., uma abstração-λ). A regra de tipagem correspondente será

$$\text{rec} \qquad \frac{y : \tau \quad t : \tau}{\mathsf{rec}\,(y = t) : \tau}$$

Por outro lado, a avaliação *lazy*, permite uma noção mais ampla de forma canónica:

- todos os valores booleanos ou inteiros são formas canónicas;
- um tuplo de termos fechados é uma forma canónica;
- todas as expressões-λ fechadas são formas canónicas;

Aula 23. [(TP) Seg, 11 Jun 2008, 11 - 13 h]

Sumário:
Conclusão do *estudo de caso*: semântica das linguagens funcionais com tipos de ordem superior sob avaliação *lazy*.

LFS++ **(com avaliação *lazy*): Semântica operacional**

$$(Lfc) \qquad c \Downarrow c \quad (c \text{ forma canónica})$$

$$(Lop) \qquad \frac{t_1 \Downarrow v_1 \quad t_2 \Downarrow v_2}{t_1\, op\, t_2 \Downarrow v} \quad \Leftarrow\ v = v_1\, op\, v_2$$

$$(Lif1) \qquad \frac{tb \Downarrow \mathsf{true} \quad t_1 \Downarrow c_1}{\mathsf{if}\ tb\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 \Downarrow c_1}$$

$$(Lif2) \qquad \frac{tb \Downarrow \mathsf{false} \quad t_2 \Downarrow c_2}{\mathsf{if}\ tb\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 \Downarrow c_2}$$

$$(Lp1) \qquad \frac{t \Downarrow (t_1, t_2) \quad t_1 \Downarrow c_1}{\mathsf{fst}\ t \Downarrow c_1}$$

$$(Lp2) \qquad \frac{t \Downarrow (t_1, t_2) \quad t_2 \Downarrow c_2}{\mathsf{snd}\ t \Downarrow c_2}$$

$$(Lfu) \qquad \frac{t_1 \Downarrow \lambda\, x.t_1' \quad t_1'[t_2/x] \Downarrow c}{(t_1\ t_2) \Downarrow c}$$

$$(Llt) \qquad \frac{t_2[t_1/x] \Downarrow c}{\mathsf{let}\ (x \leftarrow t_1)\ \mathsf{in}\ t_2 \Downarrow c}$$

$$(Lrc) \qquad \frac{t[\mathsf{rec}\,(x = t)/x]) \Downarrow c}{\mathsf{rec}\,(x = t) \Downarrow c}$$

Exercício.
Explique a razão pela qual a semântica operacional para a versão *lazy* da linguagem omite uma regra para a avaliação de pares ordenados.

Exercício.
Compare cuidadosamente as semânticas operacionais para as versões *eager* e *lazy* de LFS++. Explique todas as diferenças que encontrar.

Exercício.
Mostre que a semântica operacional para a versão *lazy* de LFS++ é determinística e respeita os tipos (comece por formular estas propriedades e recorra, depois, a indução sobre a estrutura das regras para as verificar).

LFS++ **(tipos): Semântica denotacional**

$$
\begin{aligned}
V_{int} &= \mathbb{Z} \\
V_{bool} &= \mathbb{B} \\
V_{\tau_1 * \tau_2} &= (V_{\tau_1})_\perp \times (V_{\tau_2})_\perp \\
V_{\tau_1 \longrightarrow \tau_2} &= (V_{\tau_2})_\perp^{(V_{\tau_1})_\perp}
\end{aligned}
$$

LFS++ **(termos): Semântica denotacional**

- Como usualmente os tipos contém variáveis livres, o seu significado é relativo a um ambiente de variáveis devidamente tipadas

$$Venv \;=\; \rho : Var \longrightarrow \langle \bigcup (V_\tau)_\perp \;::\; \tau \text{ é tipo} \rangle$$

- A cada termo tipado $t : \tau$ a semântica denotacional associa um elemento $[\![t]\!]\rho$ do domínio $(V_\tau)_\perp$ calculado relativamente ao ambiente de variáveis ρ.

A função semântica aplicada a um termo t do tipo τ

$$[\![t]\!] : Venv \longrightarrow (V_\tau)_\perp$$

é, assim, definida por

$$
\begin{aligned}
[\![c]\!]\,\rho &= \lfloor v \rfloor \\
[\![x]\!]\,\rho &= \rho\, x \\
[\![t_1 op\, t_2]\!]\,\rho &= [\![t_1]\!]\rho\; op_\perp\; [\![t_2]\!]\rho \\
[\![\mathsf{if}\ tb\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2]\!]\,\rho &= \begin{cases} [\![tb]\!]\rho & \Rightarrow\ [\![t_1]\!]\rho \\ \neg[\![tb]\!]\rho & \Rightarrow\ [\![t_2]\!]\rho \end{cases} \\
[\![(t_1, t_2)]\!]\,\rho &= \lfloor ([\![t_1]\!]\rho, [\![t_2]\!]\rho) \rfloor \\
[\![\mathsf{fst}(t)]\!]\,\rho &= \pi_1\, [\![t]\!]\rho \\
[\![\mathsf{snd}(t)]\!]\,\rho &= \pi_2\, [\![t]\!]\rho \\
[\![\lambda x.t]\!]\,\rho &= \lfloor \lambda\, v.[\![t]\!]\rho[v/x] \rfloor \\
[\![(t_1\, t_2)]\!]\,\rho &= let\ \phi = [\![t_1]\!]\rho\ in\ \phi([\![t_2]\!]\,\rho) \\
[\![\mathsf{let}\ (x \leftarrow t_1)\ \mathsf{in}\ t_2]\!]\,\rho &= [\![t_2]\!]\rho[[\![t_1]\!]\rho/x] \\
[\![\mathsf{rec}\ (y = t)]\!]\,\rho &= \langle \mu\ d\ ::\ [\![t]\!]\rho\,[d/x] \rangle
\end{aligned}
$$