4 Semântica Denotational

4.1 Semântica Denotacional

Semântica denotacional é um estilo de semântica que visa dar modelos matemáticos para linguagens de programação (por isso ela também é chamada *semântica matemática*). Significados de programas são definidos como sendo elementos de alguma estrutura matemática apropriada. Um comando de IMP, por exemplo, dado um estado inicial, produz um estado final, ou não termina. Podemos então dizer que comandos de IMP denotam funções parciais entre estados que, por sua vez, são representados como funções de identificadores de variáveis para inteiros.

Objetos Semânticos. Num primeiro momento é suficiente assumir que as estruturas matemáticas apropriadas para a semântica denotacional de IMP são simplesmente conjuntos.

Representação da memória (178)

- Uma memória é uma função total de identificadores para valores inteiros:

$$\sigma, \sigma', \sigma_1, \sigma_2 \ldots \in \mathsf{Ident} \to \mathbb{Z}$$

- $\Sigma \equiv \mathsf{Ident} \to \mathbb{Z}$
- $\sigma[x \mapsto n]$ é o mesmo que σ, exceto que mapeia x para n
- Notação:

$$\sigma[x \mapsto n](y) = \begin{cases} n & \text{se x=y} \\ \sigma(y) & \text{caso contrário} \end{cases}$$

Um estado (ou uma memória) é um mapeamento pertencente ao conjunto de funções $\mathsf{Ident} \to \mathbb{Z}$ e vamos assumir que essas funções são totais, ou seja que todo identificador possui um valor inteiro associado a ele na memória. Por conveniência, de agora em diante vamos nos referir ao conjunto $\mathsf{Ident} \to \mathbb{Z}$ pelo nome Σ e usamos σ, σ' ... como meta-variáveis para elementos de Σ. Portanto, para IMP temos:

Objetos Semânticos para IMP (179)

- Os significados de elementos de Aexp ?

  funções totais pertencentes a $\Sigma \to \mathbb{Z}$

- Os significados de elementos de Bexp ?

  funções totais pertencentes a $\Sigma \to \mathbb{B}$ onde $\mathbb{B} = \{\text{verdadeiro}, \text{falso}\}$

- Os significados de elementos de Com ?

  funções **parciais** pertencentes a $\Sigma \rightharpoonup \Sigma$

Exercício 4.1 Explique porque os significados de expressões aritméticas da linguagem IMP são (i) funções pertencentes a $\Sigma \to \mathbb{Z}$ e não simplesmente elementos de $\mathbb{Z}$ e (ii) porque são funções totais

Exercício 4.2 Explique porque os significados de expressões booleans da linguagem IMP são (i) funções pertencentes a $\Sigma \to \mathbb{B}$ e não simplesmente elementos de $\mathbb{B}$ e (ii) porque são funções totais.

Exercício 4.3 Explique porque os significados de comandos da linguagem IMP são funções pertencentes a $\Sigma \to \Sigma$ e (ii) porque são funções parciais.

Exercício 4.4 Como podemos *transformar* as funções que são significados de comandos da linguagem IMP em funções **totais** pertencentes a $\Sigma \to \Sigma$

Funções auxiliares. As seguintes funções serão usadas na definição da semântica denotacional de IMP: $+, -, \times, =, \leq, \neg, \vee$ e $\wedge$. Observe, que esses símbolos são sobrecarregados. O contexto de uso dirá se eles são operadores (sintáticos) da linguagem IMP ou se são os nomes das funções matemáticas.

Funções Semânticas Funções semânticas definem o mapeamento entre a sintaxe é a semântica. Em geral, temos uma função semântica para cada categoria sintática. Os nomes das funções semânticas são tradicionais em semântica denotacional.

Funções Semânticas (180)

A semântica denotacional de IMP é dada através de três funções semânticas

$$\begin{aligned} A[\![\cdot]\!] &\in \text{Aexp} \to (\Sigma \to \mathbb{Z}) \\ B[\![\cdot]\!] &\in \text{Bexp} \to (\Sigma \to \mathbb{B}) \\ C[\![\cdot]\!] &\in \text{Com} \to (\Sigma \rightharpoonup \Sigma) \end{aligned}$$

Para $a \in$ Aexp, $b \in$ Bexp e $c \in$ Com temos:

$$\begin{aligned} A[\![a]\!] &\in \Sigma \to \mathbb{Z} \\ B[\![b]\!] &\in \Sigma \to \mathbb{B} \\ C[\![c]\!] &\in \Sigma \rightharpoonup \Sigma \end{aligned}$$

E, para $\sigma \in \Sigma$:

$$\begin{array}{l} A[\![a]\!]\sigma \in \mathbb{Z} \\ B[\![b]\!]\sigma \in \mathbb{B} \\ C[\![c]\!]\sigma \in \Sigma \quad \text{caso } C[\![c]\!]\sigma \text{ for definido} \end{array}$$

Equações Semânticas A definição das funções semânticas, e portanto a definição do significado matemático de construções de IMP, consiste de um conjunto de *equações semânticas*.

Definições (181)

- As *equações semânticas* podem ser escritas na forma

$$\begin{array}{rcl} A[\![a]\!]\sigma & = & \ldots \\ B[\![b]\!]\sigma & = & \ldots \\ C[\![c]\!]\sigma & = & \ldots \end{array}$$

- ou, usando a notação lambda, na forma

$$\begin{array}{rcl} A[\![a]\!] & = & \lambda\sigma \in \Sigma. \quad \ldots \\ B[\![b]\!] & = & \lambda\sigma \in \Sigma. \quad \ldots \\ C[\![c]\!] & = & \lambda\sigma \in \Sigma. \quad \ldots \end{array}$$

Tanto $A[\![a]\!]\sigma = \ldots$ como $A[\![a]\!] = \lambda\sigma \in \Sigma. \quad \ldots$ possuem a mesma leitura:

"a denotação da expressão a é a função que, dada uma memória $\sigma \ldots$"
"a denotação da expressão b é a função que, dada uma memória $\sigma \ldots$"
"a denotação do comando c é a função que, dada uma memória $\sigma \ldots$"

Equações Semânticas para Expressões Aritméticas (182)

O significado de uma expressão aritmética é uma função em $\Sigma \to \mathbb{Z}$ definida como segue:

$$A[\![\mathbf{n}]\!]\sigma = n \tag{EqN}$$
$$A[\![\mathbf{x}]\!]\sigma = \sigma(x) \tag{EqId}$$
$$A[\![\mathbf{a_1}\texttt{+}\mathbf{a_2}]\!]\sigma = A[\![\mathbf{a_1}]\!]\sigma + A[\![\mathbf{a_2}]\!]\sigma \tag{Eq+}$$
$$A[\![\mathbf{a_1}\texttt{-}\mathbf{a_2}]\!]\sigma = A[\![\mathbf{a_1}]\!]\sigma - A[\![\mathbf{a_2}]\!]\sigma \tag{Eq-}$$
$$A[\![\mathbf{a_1}\texttt{*}\mathbf{a_2}]\!]\sigma = A[\![\mathbf{a_1}]\!]\sigma \times A[\![\mathbf{a_2}]\!]\sigma \tag{Eq*}$$

Exercício:

$$\begin{array}{rcll} A[\![\texttt{x + (5 * 4)}]\!]\sigma & = & A[\![\texttt{x}]\!]\sigma + A[\![5 * 4]|\sigma & (Eq+) \\ & = & \sigma(\mathbf{x}) + A[\![5 * 4]|\sigma & (EqId) \\ & = & \sigma(\mathbf{x}) + A[\![5]|\sigma \times A[\![4]|\sigma & (Eq*) \\ & = & \sigma(\mathbf{x}) + (5 \times 4) & (EqN, EqN) \\ & = & \sigma(\mathbf{x}) + 20 & \end{array}$$

Note que a definição do significado de expressões booleanas faz uso da função de atribuição de significado para expressões aritméticas:

Equações Semânticas para Expressões Boolenas (183)

$$
\begin{aligned}
&B[\![\mathbf{t}]\!]\sigma && = t && \text{(EqB)}\\
&B[\![\mathbf{a_1}\texttt{=}\mathbf{a_2}]\!]\sigma && = A[\![\mathbf{a_1}]\!]\sigma = A[\![\mathbf{a_2}]\!]\sigma && \text{(Eq=)}\\
&B[\![\mathbf{a_1}\texttt{<}\mathbf{a_2}]\!]\sigma && = A[\![\mathbf{a_1}]\!]\sigma < A[\![\mathbf{a_2}]\!]\sigma && \text{(Eq<)}\\
&B[\![\neg\mathbf{b}]\!]\sigma && = \neg B[\![\mathbf{b}]\!]\sigma && \text{(EqNeg)}\\
&B[\![\mathbf{b_1}\wedge\mathbf{b_2}]\!]\sigma && = B[\![\mathbf{b_1}]\!]\sigma \wedge B[\![\mathbf{b_2}]\!]\sigma && \text{(EqAnd)}\\
&B[\![\mathbf{b_1}\vee\mathbf{b_2}]\!]\sigma && = B[\![\mathbf{b_1}]\!]\sigma \vee B[\![\mathbf{b_2}]\!]\sigma && \text{(EqOr)}
\end{aligned}
$$

Segue abaixo a definição da denotação de comandos. O significado do comando while exige cuidados e será visto mais adiante:

Equações Semânticas para Comandos (184)

O significado de um comando é uma função em $\Sigma \rightharpoonup \Sigma$ definida como segue

$$
\begin{aligned}
C[\![\texttt{skip}]\!]\sigma &= \sigma && \text{(EqSkp)}\\
C[\![\mathbf{x}\texttt{:=}\mathbf{a}]\!]\sigma &= \sigma[\mathbf{x} \mapsto A[\![\mathbf{a}]\!]\sigma] && \text{(EqAtr)}\\
C[\![\mathbf{c_1}\texttt{;}\mathbf{c_2}]\!]\sigma &= C[\![\mathbf{c_2}]\!](C[\![\mathbf{c_1}]\!]\sigma) && \text{(EqSeq)}\\
C[\![\texttt{if } \mathbf{b} \texttt{ then } \mathbf{c_1} \texttt{ else } \mathbf{c_2}]\!]\sigma &= \textit{se } B[\![\mathbf{b}]\!]\sigma \textit{ então } C[\![\mathbf{c_1}]\!]\sigma \textit{ senão } C[\![\mathbf{c_2}]\!]\sigma && \text{(EqIf)}\\
C[\![\texttt{while } \mathbf{b} \texttt{ do } \mathbf{c}]\!] &= \ldots && \text{(EqWh)}
\end{aligned}
$$

Segue abaixo um exemplo no qual é calculado o significado matemático de um comando utilizando as equações acima:

Exemplo (185)

$$
\begin{aligned}
\underline{C[\![\texttt{x:=1;(skip;y:=2)}]\!]\sigma} &= C[\![\texttt{skip;y:=2}]\!]\ (\underline{C[\![\texttt{x:=1}]\!]\sigma}) && \text{(EqSeq)}\\
&= C[\![\texttt{skip;y:=2}]\!]\ \sigma[\mathbf{x} \mapsto \underline{A[\![1]\!]\sigma}] && \text{(EqAtr)}\\
&= \underline{C[\![\texttt{skip;y:=2}]\!]\ \sigma[\mathbf{x} \mapsto 1]} && \text{(EqN)}\\
&= C[\![\texttt{y:=2}]\!]\ (\underline{C[\![\texttt{skip}]\!]\sigma[\mathbf{x} \mapsto 1]}) && \text{(EqSeq)}\\
&= \underline{C[\![\texttt{y:=2}]\!]\ \sigma[\mathbf{x} \mapsto 1]} && \text{(EqSkp)}\\
&= \sigma[\mathbf{x} \mapsto 1][\mathbf{y} \mapsto \underline{A[\![2]\!]\sigma[\mathbf{x} \mapsto 1]}] && \text{(EqAtr)}\\
&= \sigma[\mathbf{x} \mapsto 1][\mathbf{y} \mapsto 2] && \text{(EqN)}
\end{aligned}
$$

Semântica com funções totais. O tratamento formal fica mais simples se lidar somente com funções totais. Podemos adicionar o elemento $\perp$ (lido "bottom", ou "indefinido") ao conjunto Σ para representar indefinição. Escrevemos $\Sigma_\perp$ para nomear o conjunto $\Sigma \cup \{\perp\}$. Ao invés de dizermos que os significados de comandos pertencem ao conjunto de funções parciais de Σ para Σ dizemos que pertencem ao conjunto de funções totais de Σ para $\Sigma_\perp$. E, finalmente, como o comando $\mathbf{c_1}$ em $\mathbf{c_1};\mathbf{c_2}$ pode não terminar, dizemos que os significados de comandos pertencem ao conjunto de funções totais $\Sigma_\perp \to \Sigma_\perp$. Essas funções são *estritas*, ou seja se aplicadas a $\perp$ produzem $\perp$.

***Lifting* - trabalhando com funções totais (186)**

- $\Sigma_\perp = \Sigma \cup \{\perp\}$
- $\perp$ representa a indefinição
- $C[\![\texttt{c}]\!]$ é agora função **total** em $\Sigma_\perp \to \Sigma_\perp$
- Exemplo :

$$
\begin{aligned}
C[\![(\texttt{while true do c}); c_2]\!]\sigma &= C[\![c_2]\!](\underline{C[\![\texttt{while true do c}]\!]\sigma})\\
&= C[\![c_2]\!]\perp\\
&= \perp
\end{aligned}
$$

Significado do Comando `while` Como fica a equação para o comando while? Podemos começar lembrando que o significado do comando `while b do c` é equivalente ao significado do comando

`if` **b** `then (`**c**`;while` **b** `do` **c**`) else skip`.

Logo

Significado do comando while? (187)

$$
\begin{aligned}
C[\![\texttt{while}\ \mathbf{b}\ \texttt{do}\ \mathbf{c}]\!]\sigma\ &= \underline{C[\![\texttt{if}\ \mathbf{b}\ \texttt{then}\ (\mathbf{c};\texttt{while}\ \mathbf{b}\ \texttt{do}\ \mathbf{c})\ \texttt{else}\ \texttt{skip}]\!]\sigma} \\
&= \mathit{se}\ B[\![\mathbf{b}]\!]\sigma\ \mathit{então}\ \underline{C[\![\mathbf{c};\texttt{while}\ \mathbf{b}\ \texttt{do}\ \texttt{c}]\!]\sigma}\ \mathit{senão}\ C[\![\texttt{skip}]\!]\sigma && (\mathit{EqIf}) \\
&= \mathit{se}\ B[\![\mathbf{b}]\!]\sigma\ \mathit{então}\ C[\![\texttt{while}\ \mathbf{b}\ \texttt{do}\ \texttt{c}]\!](C[\![\mathbf{c}]\!]\sigma)\ \mathit{senão}\ \underline{C[\![\texttt{skip}]\!]\sigma} && (\mathit{EqSeq}) \\
&= \mathit{se}\ B[\![\mathbf{b}]\!]\sigma\ \mathit{então}\ C[\![\texttt{while}\ \mathbf{b}\ \texttt{do}\ \texttt{c}]\!](C[\![\mathbf{c}]\!]\sigma)\ \mathit{senão}\ \sigma. && (\mathit{EqSkp})
\end{aligned}
$$

Ou seja:

$$[\![\texttt{while}\ \mathbf{b}\ \texttt{do}\ \mathbf{c}]\!]\sigma \ = \ \text{se } [\![\mathbf{b}]\!]\sigma \text{ então } [\![\texttt{while}\ \mathbf{b}\ \texttt{do}\ \texttt{c}]\!]([\![\mathbf{c}]\!]\sigma) \text{ senão } \sigma$$

ou, usando $\mathbf{W}$ para $[\![\texttt{while}\ \mathbf{b}\ \texttt{do}\ \mathbf{c}]\!]$:

$$\mathbf{W}\,\sigma \ = \ \text{se } [\![\mathbf{b}]\!]\sigma \text{ se } \mathbf{W}([\![\mathbf{c}]\!]\sigma) \text{ senão } \sigma$$

Contudo a equação

$$C[\![\texttt{while}\ \mathbf{b}\ \texttt{do}\ \mathbf{c}]\!]\sigma = \mathit{se}\ B[\![\mathbf{b}]\!]\sigma\ \mathit{então}\ C[\![\texttt{while}\ \mathbf{b}\ \texttt{do}\ \mathbf{c}]\!](C[\![\mathbf{c}]\!]\sigma)\ \mathit{senão}\ \sigma \quad (\mathit{EqWhV1})$$

não é suficiente para definir o significado de comandos `while`, e a razão é simples: ao contrário das demais equações para a linguagem – que definem explicitamente uma única função de $\Sigma_\perp \to \Sigma_\perp$ dado um comando – está equação (como veremos nos exercícios abaixo), dado um comando while específico, pode ter várias soluções. O significado do comando é, com certeza, uma dessas soluções, mas a equação acima, por si só, não indica qual delas.

Equações e Definições - Revisão (188)

- Dado um universo U, uma equação define um elemento de U quando ela
  - possui solução em U, e
  - a solução é única
- Se uma equação possui mais de uma solução, ela não define um elemento de U
- Uma propriedade adicional pode ser adotada para apontar uma das soluções como sendo o elemento definido

Equações e Definições - Revisão (189)

- Exemplos:
  - $x = x + 1$ não possui solução, seja lá qual for o universo
  - $x = 3x - 4$ possui o natural 2 como uma única solução em N, logo essa equação define x como sendo o natural 2

– $2x^2 = 8$ possui duas soluções em R: 2 e -2, logo ela por si só não define nenhum número real

Exercício 4.5 Considere o comando `while x` $\neq$ `0 do x:=x-2`

- Obtenha a equação associada a esse comando

  No que segue $\mathbf{W} \equiv [\![\texttt{while } (x \neq 0) \texttt{ do } x := x - 2]\!]$:

$$\begin{array}{rcll}
\mathbf{W}\sigma & = & \textit{se } \underline{[\![x \neq 0]\!]\sigma} \textit{ então } \mathbf{W}([\![x := x-2]\!]\sigma) \textit{ senão } \sigma & \text{(EqWhV1)} \\
 & = & \textit{se } \underline{[\![x]\!]\sigma} \neq [\![0]\!]\sigma \textit{ então } \mathbf{W}([\![x := x-2]\!]\sigma) \textit{ senão } \sigma & \text{(EqDif)} \\
 & = & \textit{se } \sigma(x) \neq \underline{[\![0]\!]\sigma} \textit{ então } \mathbf{W}([\![x := x-2]\!]\sigma) \textit{ senão } \sigma & \text{(EqId)} \\
 & = & \textit{se } \sigma(x) \neq 0 \textit{ então } \mathbf{W}(\underline{[\![x := x-2]\!]\sigma}) \textit{ senão } \sigma & \text{(EqN)} \\
 & = & \textit{se } \sigma(x) \neq 0 \textit{ então } \mathbf{W}(\sigma[x \mapsto \underline{[\![x-2]\!]\sigma}]) \textit{ senão } \sigma & \text{(EqAtr)} \\
 & = & \textit{se } \sigma(x) \neq 0 \textit{ então } \mathbf{W}(\sigma[x \mapsto \underline{[\![x]\!]\sigma} - [\![2]\!]\sigma]) \textit{ senão } \sigma & \text{(EqSub)} \\
 & = & \textit{se } \sigma(x) \neq 0 \textit{ então } \mathbf{W}(\sigma[x \mapsto \sigma(x) - \underline{[\![2]\!]\sigma}]) \textit{ senão } \sigma & \text{(EqId)} \\
 & = & \textit{se } \sigma(x) \neq 0 \textit{ então } \mathbf{W}(\sigma[x \mapsto \sigma(x) - 2]) \textit{ senão } \sigma & \text{(EqN)}
\end{array}$$

  Usando $\sigma' \equiv \sigma[x \mapsto \sigma(x) - 2]$ a equação fica:

$$\mathbf{W}\sigma = \textit{se } \sigma(x) \neq 0 \textit{ então } \mathbf{W}\sigma' \textit{ senão } \sigma$$

- Verifique se as funções g_1 e g_2 definidas abaixo são soluções dessa equação:

$$g_1\sigma = \begin{cases} \sigma[x \mapsto 0] & \text{se } \sigma(x) \text{ é par e } \sigma(x) \geq 0 \\ \bot & \text{caso contrário} \end{cases}$$

$$g_2\sigma = \begin{cases} \sigma[x \mapsto 0] & \text{se } \sigma(x) \text{ é par e } \sigma(x) \geq 0 \\ \sigma_1 & \text{se } \sigma(x) \text{ é par e } \sigma(x) < 0 \\ \sigma_2 & \text{se } \sigma(x) \text{ é ímpar} \end{cases}$$

  onde σ_1 e σ_2 podem ser estados quaisquer ou $\bot$

  Verificação se g_1 é solução da equação acima:

$$\begin{aligned}
g_1\sigma &= \textit{se } \sigma(x) \neq 0 \textit{ então } \underline{g_1\sigma'} \textit{ senão } \sigma \\
&= \textit{se } \sigma(x) \neq 0 \textit{ então } (\textit{se par}(\underline{\sigma'(x)}) \wedge \sigma'(x) \geq 0 \textit{ então } \sigma'[x \mapsto 0] \textit{ senão } \bot) \textit{ senão } \sigma \\
&= \textit{se } \sigma(x) \neq 0 \textit{ então } (\textit{se } \underline{\textit{par}(\sigma(x) - 2)} \wedge \sigma'(x) \geq 0 \textit{ então } \sigma'[x \mapsto 0] \textit{ senão } \bot) \textit{ senão } \sigma \\
&= \textit{se } \sigma(x) \neq 0 \textit{ então } (\textit{se par}(\sigma(x)) \wedge \underline{\sigma'(x)} \geq 2 \textit{ então } \sigma'[x \mapsto 0] \textit{ senão } \bot) \textit{ senão } \sigma \\
&= \textit{se } \sigma(x) \neq 0 \textit{ então } (\textit{se par}(\sigma(x)) \wedge \underline{\sigma(x) - 2 \geq 0} \textit{ então } \sigma[x \mapsto 0] \textit{ senão } \bot) \textit{ senão } \sigma \\
&= \textit{se } \sigma(x) \neq 0 \textit{ então } (\textit{se par}(\sigma(x)) \wedge \sigma(x) \geq 2 \textit{ então } \sigma[x \mapsto 0] \textit{ senão } \bot) \textit{ senão } \underline{\sigma} \\
&= \textit{se } \sigma(x) \neq 0 \textit{ então } (\textit{se par}(\sigma(x)) \wedge \sigma(x) \geq 2 \textit{ então } \sigma[x \mapsto 0] \textit{ senão } \bot) \textit{ senão } \sigma[x \mapsto 0] \\
&= \textit{se par}(\sigma(x)) \wedge \sigma(x) \geq 0 \textit{ então } \sigma[x \mapsto 0] \textit{ senão } \bot \\
&= g_1\sigma
\end{aligned}$$

Verificação se g_2 é solução da equação acima:

$$\begin{aligned}
g_2\sigma &= \textit{se } \sigma(x) \neq 0 \textit{ então } \underline{g_2\sigma'} \textit{ senão } \sigma \\
&= \textit{se } \sigma(x) \neq 0 \textit{ entao } (\textit{se par}(\sigma'(x)) \wedge \sigma'(x) \geq 0 \textit{ entao } \sigma'[x \mapsto 0] \textit{ senao} \\
&\qquad\qquad\qquad \textit{se par}(\sigma'(x)) \wedge \sigma'(x) < 0 \textit{ entao } \sigma_2 \textit{ senao } \sigma_3) \\
&\quad \textit{senao } \sigma \\
&= \textit{se } \sigma(x) \neq 0 \textit{ entao } (\textit{se par}(\sigma(x)) \wedge \sigma(x) \geq 2 \textit{ entao } \sigma[x \mapsto 0] \textit{ senao} \\
&\qquad\qquad\qquad \textit{se par}(\sigma(x)) \wedge \sigma(x) < 2 \textit{ entao } \sigma_2 \textit{ senao } \sigma_3) \\
&\quad \textit{senao } \sigma[x \mapsto 0] \\
&= \textit{se par}(\sigma(x)) \wedge \sigma(x) \geq 0 \textit{ entao } \sigma[x \mapsto 0] \textit{ senao} \\
&\quad \textit{se par}(\sigma(x)) \wedge \sigma(x) < 0 \textit{ entao } \sigma_2 \textit{ senao } \sigma_3) \\
&= g_2\sigma
\end{aligned}$$

- Qual das duas funções, g_1 ou g_2 deve ser o significado do comando while acima?

Exercício 4.6 Considere o comando `while true do skip`.

- Escreva a equação associada ao comando e explique por que **todas** as funções em $\Sigma_\bot \to \Sigma_\bot$ são soluções para esta equação.
- Qual dessas funções é o significado deste comando?

Exercício 4.7 Considere o comando `while ¬(x=0) do skip`.

- Escreva a equação correspondente ao comando while acima;

- Verifique se as funções g_1 e g_2 abaixo são soluções desta equação

$$g_1\sigma = \begin{cases} \bot & \text{se } \sigma(x) \neq 0 \\ \sigma & \text{se } \sigma(x) = 0 \end{cases}$$
$$g_2\sigma = \begin{cases} \sigma_3 & \text{se } \sigma(x) \neq 0 \\ \sigma & \text{se } \sigma(x) = 0 \end{cases}$$

- Qual dessas funções é o significado do comando?

- Certifique-se de que a função g definida abaixo não é solução da equação acima

$$g\sigma = \bot \qquad \text{para todo } \sigma \in \Sigma$$

Exercício 4.8 Considere o comando `while ¬(x=0) do x:=x-1`.

- Escreva a equação associada a esse comando.

- Verifique se funções definidas abaixo são soluções dessa equação:

$$g_1\sigma = \bot \qquad \text{para todo } \sigma \in \Sigma$$
$$g_2\sigma = \begin{cases} \sigma[x \mapsto 0] & \text{se } \sigma x \geq 0 \\ \bot & \text{se } \sigma x < 0 \end{cases}$$
$$g_3\sigma = \begin{cases} \sigma[x \mapsto 0] & \text{se } \sigma x \geq 0 \\ \sigma & \text{se } \sigma x < 0 \end{cases}$$
$$g_4\sigma = \sigma[x \mapsto 0] \qquad \text{para todo } \sigma \in \Sigma$$
$$g_5\sigma = \sigma \qquad \text{para todo } \sigma \in \Sigma$$

- Qual dessa funções é o significado do comando while acima?

4.2 O significado é a menor solução

Observe, pelos exercícios anteriores, que há uma relação de ordem entre as soluções de uma equação baseada na "quantidade" de informação que cada uma expressa. Formalmente podemos definir essa relação de ordem da seguinte maneira:

Definição de *menor* (190)

- Considere o seguinte relação ordem $\sqsubseteq$ entre funções de$\Sigma_\bot \to \Sigma_\bot$:

$$f \sqsubseteq g \iff \forall \sigma\, f\sigma = \bot \vee f\sigma = g\sigma \tag{4.1}$$

- Nessa ordem uma função com "menos informação" é dita *menor*.

ou seja, quando f é definida, g também é definida e produz o mesmo resultado do que f. Em todos exemplos acima, a solução correspondente ao significado do comando sempre foi a menor solução (dada a relação de ordem acima).

Formulamos então a seguinte hipótese:

Hipótese - o significado é a menor solução (191)

o significado de um comando while é a menor solução da equação

$$C[\![\texttt{while } \mathbf{b} \texttt{ do } \mathbf{c}]\!]\sigma = \mathit{se}\ B[\![\texttt{b}]\!]\sigma\ \mathit{então}\ C[\![\texttt{while } \mathbf{b} \texttt{ do } \texttt{c}]\!](C[\![\mathbf{c}]\!]\sigma)\ \mathit{senão}\ \sigma$$

Exercício 4.9 Ordene as soluções das equações dos exercícios anteiores utilizando a definição de ordem dada acima.

4.3 Equações de Ponto Fixo

Até agora, usando somente a nossa intuição e alguns exemplos, levantamos a hipótese de que o significado de comando `while` deve ser a menor solução da equação para o comando while associada associada. Mas como obter/calcular essa menor solução?

Para começar vai ser necessário colocar a equação *(EqWhV1)* no formato geral de equação de ponto fixo. Equações de ponto fixo são aquelas que podem ser colocadas no seguinte formato geral $x = f(x)$. Uma solução para uma equação de ponto fixo $x = f(x)$ é dita ponto fixo de f.

Equação no formato de equação de ponto fixo (192)

A equação

$$C[\![\texttt{while } \mathbf{b} \texttt{ do } \mathbf{c}]\!]\sigma = \mathit{se}\ B[\![\texttt{b}]\!]\sigma\ \mathit{então}\ C[\![\texttt{while } \mathbf{b} \texttt{ do } \texttt{c}]\!](C[\![\mathbf{c}]\!]\sigma)\ \mathit{senão}\ \sigma$$

no formato geral de equação de ponto fixo, fica:

$$[\![\texttt{while}\ \ b\ do\ c]\!] = F\ ([\![\texttt{while}\ b\ do\ c]\!])$$

$$\text{onde}\quad F \in (\Sigma_\perp \to \Sigma_\perp) \to (\Sigma_\perp \to \Sigma_\perp)\ \text{é definido como:}$$

$$Fg = \lambda\sigma \in \Sigma_\perp.\ se\ [\![b]\!]\sigma\ então\ g([\![c]\!]\sigma)\ senão\ \sigma$$

Sendo assim a nossa hipótese a cerca do significado de qualquer comando while pode ser reescrita nas seguintes formas equivalentes:

Hipótese - o significado é a menor solução (II) (193)

- O significado de um comando while é a menor solução da equação abaixo, ou ainda
- O significado de um comando while é o menor ponto fixo de F da equação abaixo

$$[\![\texttt{while}\ \ b\ c]\!] = F\ ([\![\texttt{while}\ b\ c]\!])$$

onde: $F \in (\Sigma_\perp \to \Sigma_\perp) \to (\Sigma_\perp \to \Sigma_\perp)$ *é definido como:*

$$Fg = \lambda\sigma \in \Sigma.\ se\ [\![b]\!]\sigma\ então\ g([\![c]\!]\sigma)\ senão\ \sigma$$

Outra forma equivalente de formular a nossa hipótese faz uso da seguinte notação empregada para se referir ao menor ponto fixo de F: `fix` F ou ainda, $\mu\ F$

Hipótese (III) (194)

$$[\![\texttt{while}\ \ b\ c]\!] = \texttt{fix}\ F$$

onde: $F \in (\Sigma_\perp \to \Sigma_\perp) \to (\Sigma_\perp \to \Sigma_\perp)$ *é definido como:*

$$Fg = \lambda\sigma \in \Sigma.\ se\ [\![b]\!]\sigma\ então\ g([\![c]\!]\sigma)\ senão\ \sigma$$

Exercício 4.10 Reescreva as equações dos comandos while dos exemplos/exercícios das seções anteriores colocando-as no formato geral de equação de ponto fixo.

4.4 Calculando menor ponto fixo

Para obter o menor ponto fixo de F começamos aplicando F a função totalmente indefinida de $\Sigma_\perp \to \Sigma_\perp$ e assim aplicamos F sucessivamente a cada resultado obtido anteriormente. A idéia é que, acada aplicação de F obtemos uma função de $\Sigma_\perp \to \Sigma_\perp$ com mais informação e, eventualmente, obtemos uma função que é ponto fixo de F (ou seja que é o significado do comando while).

Exemplo 1 (195)

Considere `while x` $\neq$ `0 do x:=x-2`. O funcional F associado a esse comando é:

$$F\ g = \lambda\sigma \in \Sigma.\ \text{ se } \sigma(x) \neq 0 \text{ então } g\sigma[x \mapsto \sigma(x) - 2] \text{ senão } \sigma$$

Usando σ' no lugar de $\sigma[x \mapsto \sigma(x) - 2]$ temos:

$$F\ g = \lambda\sigma \in \Sigma.\ \text{ se } \sigma(x) \neq 0 \text{ então } g\sigma' \text{ senão } \sigma$$

Calculando $F^0(\bot)$, $F^1(\bot)$, $F^2(\bot)$, ...:

Exemplo 1 (196)

$$F^0(\bot) =_{\text{def}} \bot$$

$$\begin{aligned}
F^1(\bot) &= F(\bot)\\
&= \lambda\sigma \in \Sigma.\ \text{se}\ \sigma(x) \neq 0\ \text{então}\ \bot(\sigma')\ \text{senão}\ \sigma\\
&= \lambda.\sigma \in \Sigma.\ \text{se}\ \sigma(x) \neq 0\ \text{então}\ \bot_{\Sigma_\bot}\ \text{senão}\ \sigma[x \mapsto 0]\\
&= \lambda.\sigma \in \Sigma.\ \text{se}\ \sigma(x) = 0\ \text{então}\ \sigma[x \mapsto 0]\ \text{senão}\ \bot_{\Sigma_\bot}
\end{aligned}$$

$$\begin{aligned}
F^2(\bot) &= F(F^1(\bot))\\
&= \lambda\sigma \in \Sigma.\ \text{se}\ \sigma(x) \neq 0\ \text{então}\ (F^1\bot)\sigma'\ \text{senão}\ \sigma\\
&= \lambda\sigma \in \Sigma.\ \text{se}\ \sigma(x) \neq 0\ \text{então}\ (\text{se}\ \sigma'(x) = 0\ \text{então}\ \sigma'[x \mapsto 0]\ \text{senão}\ \bot_{\Sigma_\bot})\ \text{senão}\ \sigma\\
&= \lambda\sigma \in \Sigma.\ \text{se}\ \sigma(x) \neq 0\ \text{então}\ (\text{se}\ \sigma(x) = 2\ \text{então}\ \sigma[x \mapsto 0]\ \text{senão}\ \bot_{\Sigma_\bot})\ \text{senão}\ \sigma\\
&= \lambda\sigma \in \Sigma.\ \text{se}\ \sigma(x) = 0 \vee \sigma(x) = 2\ \text{então}\ \sigma[x \mapsto 0]\ \text{senão}\ \bot_{\Sigma_\bot}
\end{aligned}$$

$$\begin{aligned}
F^3(\bot) &= F(F^2(\bot))\\
&= \ldots\\
&= \lambda\sigma \in \Sigma.\ \text{se}\ \sigma(x) = 0 \vee \sigma(x) = 2 \vee \sigma(x) = 4\ \text{então}\ \sigma[x \mapsto 0]\ \text{senão}\ \bot_{\Sigma_\bot}
\end{aligned}$$

...

Exemplo 1 (197)

- Com cada aplicação de F a "informação" cresce e temos a seguinte cadeia:

$$F^0(\bot) \sqsubseteq F^1(\bot) \sqsubseteq F^2(\bot) \sqsubseteq F^3(\bot) \sqsubseteq \cdots$$

- Intuitivamente, no "limite" da cadeia acima temos

$$g = \lambda\sigma \in \Sigma. \begin{cases} \sigma[x \mapsto 0] & \text{se}\ par(\sigma(x)) \wedge x \geq 0\\ \bot & \text{caso contrário} \end{cases}$$

  representa a função que
  - é ponto fixo de F (verifique isso)
  - corresponde ao significado do comando while

Exemplo 2 (198)

Considere `while x>0 do x:=x-1`. O funcional F associado a esse comando é o seguinte:

$$F\ g = \lambda\sigma \in \Sigma.\ \text{se } \sigma(x) > 0 \text{ então } g\sigma[x \mapsto \sigma(x) - 1] \text{ senão } \sigma$$

Usando σ' para $\sigma[x \mapsto \sigma(x) - 1]$ temos:

$$F\ g = \lambda\sigma \in \Sigma.\ \text{se } \sigma(x) > 0 \text{ então } g\sigma' \text{ senão } \sigma$$

Exemplo 2 (199)

$$F^0(\perp) =_{\text{def}} \perp$$

$$\begin{aligned} F^1(\perp) &= F(\perp) \\ &= \lambda\sigma \in \Sigma.\ \text{se } \sigma(x) > 0 \text{ então } \perp\sigma' \text{ senão } \sigma \\ &= \lambda.\sigma \in \Sigma.\ \text{se } \sigma(x) > 0 \text{ então } \perp_{\Sigma_\perp} \text{ senão } \sigma \end{aligned}$$

$$\begin{aligned} F^2(\perp) &= F(F^1(\perp)) \\ &= \lambda\sigma \in \Sigma.\ \text{se } \sigma(x) > 0 \text{ então } (F^1\perp)\sigma' \text{ senão } \sigma \\ &\dots \\ &= \lambda\sigma \in \Sigma. \begin{cases} \sigma & \text{se } \sigma(x) \le 0 \\ \sigma[x \mapsto 0] & \text{se } \sigma(x) = 1 \\ \perp_{\Sigma_\perp} & \text{se } \sigma(x) > 1 \end{cases} \end{aligned}$$

$$F^3(\perp) = F(F^2(\perp)) = \cdots = \lambda\sigma \in \Sigma. \begin{cases} \sigma & \text{se } \sigma(x) \le 0 \\ \sigma[x \mapsto 0] & \text{se } \sigma(x) = 1 \vee \sigma(x) = 2 \\ \perp_{\Sigma_\perp} & \text{se } \sigma(x) > 2 \end{cases}$$

$$F^4(\perp) = F(F^3(\perp)) = \cdots = \lambda\sigma \in \Sigma. \begin{cases} \sigma & \text{se } \sigma(x) \le 0 \\ \sigma[x \mapsto 0] & \text{se } \sigma(x) = 1 \vee \sigma(x) = 2 \vee \sigma(x) = 3 \\ \perp_{\Sigma_\perp} & \text{se } \sigma(x) > 3 \end{cases}$$

Exemplo 2 (200)

- Com cada aplicação de F a “informação” cresce:

$$F^0(\perp) \sqsubseteq F^1(\perp) \sqsubseteq F^2(\perp) \sqsubseteq F^3(\perp) \sqsubseteq \cdots$$

- Intuitivamente, no "limite" da cadeia acima, temos a seguinte função:

$$\lambda\sigma \in \Sigma. \begin{cases} \sigma & \text{se } \sigma(x) \leq 0 \\ \sigma[x \mapsto 0] & \text{se } \sigma(x) > 0 \end{cases}$$

  representa a função que
  - é ponto fixo de F (verifique isso)
  - corresponde ao significado do comando while

Exemplo 3 (201)

Considere o comando `while x>0 do skip`. O funcional F associado a esse comando é o seguinte

$$Fg = \lambda\sigma \in \Sigma_\perp.\ se\ \sigma x > 0\ então\ g\sigma\ senão\ \sigma$$

Calculando $F^0(\perp)$, $F^1(\perp)$, $F^2(\perp)$, ...

$$F^0(\perp) =_{\text{def}} \perp$$

$$\begin{aligned} F^1(\perp) &= F(\perp) \\ &= \lambda\sigma \in \Sigma.\ \text{se } \sigma(x) > 0 \text{ então } \perp(\sigma) \text{ senão } \sigma \\ &= \lambda\sigma \in \Sigma.\ \text{se } \sigma(x) > 0 \text{ então } \perp_\Sigma \text{ senão } \sigma \end{aligned}$$

$$\begin{aligned} F^2(\perp) &= F(F^1(\perp)) \\ &= \lambda\sigma \in \Sigma.\ \text{se } \sigma(x) > 0 \text{ então } (F^1(\perp))(\sigma) \text{ senão } \sigma \\ &= \lambda\sigma \in \Sigma.\ \text{se } \sigma(x) > 0 \text{ então } (\text{se } \sigma(x) > 0 \text{ então } \perp_\Sigma \text{ senão } \sigma) \ \ \text{senão } \sigma \\ &= \lambda\sigma \in \Sigma.\ \text{se } \sigma(x) > 0 \text{ então } \perp_\Sigma \text{ senão } \sigma \end{aligned}$$

$$\begin{aligned} F^3(\perp) &= F(F^2(\perp)) \\ &= \ldots \\ &= \lambda\sigma \in \Sigma.\ \text{se } \sigma(x) > 0 \text{ então } \perp_\Sigma \text{ senão } \sigma \end{aligned}$$

$$\begin{aligned} F^4(\perp) &= F(F^3(\perp)) \\ &= \ldots \\ &= \lambda\sigma \in \Sigma.\ \text{se } \sigma(x) > 0 \text{ então } \perp_\Sigma \text{ senão } \sigma \end{aligned}$$

...

Exemplo 3 - continuação (202)

- Neste caso obtemos um ponto fixo para F já na terceira aplicação.
- Observe que a função

$$\lambda\sigma \in \Sigma_\perp.\ \text{se } \sigma(x) > 0 \textit{ então } \perp_\Sigma \textit{ senão } \sigma$$

  obtida dessa forma :
  - é o menor dentre os pontos fixos de F
  - corresponde ao significado do comando `while x>0 do skip`

Tendo em vista os exemplos anteriores podemos refinar a nossa hipótese sobre o significado do comando while para

$$C[\![\texttt{while}\ \mathbf{b}\ \texttt{do}\ \mathbf{c}]\!] = \text{limite superior de cadeia } F^0(\perp) \sqsubseteq F^1(\perp) \sqsubseteq F^2(\perp) \sqsubseteq F^3(\perp) \sqsubseteq \cdots$$

Observação: os termos *limite superior* e *cadeia* serão precisamente definidos mais adiante. Veremos também que uma cadeia pode ter mais do que um limite superior e que estamos interessados no **menor limite superior**. Logo

$$C[\![\texttt{while}\ \mathbf{b}\ \texttt{do}\ \mathbf{c}]\!] = \text{menor limite superior de cadeia } F^0(\perp) \sqsubseteq F^1(\perp) \sqsubseteq F^2(\perp) \sqsubseteq F^3(\perp) \sqsubseteq \cdots$$

De agora em diante vamos escrever

$$\bigsqcup_{n\geq 0}^{\infty} F^n(\perp)$$

para *menor limite superior de cadeia* $F^0(\perp) \sqsubseteq F^1(\perp) \sqsubseteq F^2(\perp) \sqsubseteq F^3(\perp) \sqsubseteq \cdots$. Assim reescrevemos a hipótese acima para

$$C[\![\texttt{while}\ \mathbf{b}\ \texttt{do}\ \mathbf{c}]\!] = \bigsqcup_{n\geq 0}^{\infty} F^n(\perp)$$

Exercício 4.11 Calcule o significado dos seguintes comandos:

- `while x≠ 0 do x:= x - 1`
- `while ¬ (x = 0) do x:=x - 1`
- `while true do skip`

4.5 Existência e unicidade

Existência e Unicidade (203)

Vimos, através de alguns exemplos, que

$$C[\![\texttt{while}\ \mathbf{b}\ \texttt{do}\ \mathbf{c}]\!] = \bigsqcup_{n\geq 0}^{\infty} F^n(\bot)$$

Naqueles exemplos foi possível mostrar também que

- $F^0\bot$, $F^1\bot$, $F^2\bot\ldots$ formavam cadeia
- $\bigsqcup_{n\geq 0}^{\infty} F^n(\bot)$ é ponto fixo de F
- $\bigsqcup_{n\geq 0}^{\infty} F^n(\bot)$ é **menor** ponto fixo de F

O objetivo dessa seção é simples: queremos ter mais garantias em relação a nossa hipótese, mais precisamente

Existência e Unicidade (204)

Precisamos mostrar que nossa hipótese é válida para qualquer comando while e não somente para aqueles dos exemplos.
Para isso primeiro provamos **existência** de $\bigsqcup_{n\geq 0}^{\infty} F^n(\bot)$, para isso temos que mostrar que

- $F^0\bot$, $F^1\bot$, $F^2\bot\ldots$ *sempre* formam cadeia $F^0\bot \sqsubseteq F^1\bot \sqsubseteq F^2\bot\ldots$ e que
- toda cadeia assim formada possui menor limite superior

A seguir, mostramos que

- $\bigsqcup_{n\geq 0}^{\infty} F^n(\bot)$ é um ponto fixo de F e
- esse ponto fixo é o **menor**

Somente depois de provada a existência e a unicidade desse menor ponto fixo de F e depois de provado que ele é igual a $\bigsqcup_{n\geq 0}^{\infty} F^n(\bot)$ é que poderemos afirmar que

$$C[\![\texttt{while}\ \mathbf{b}\ \texttt{do}\ \mathbf{c}]\!] = \bigsqcup_{n\geq 0}^{\infty} F^n(\bot)$$

Para isso vamos precisar de uma série de definições, algums resultado peliminares, culminando com a prova do Teorema do Ponto Fixo.

4.5.1 $(\Sigma_\bot \to \Sigma_\bot, \sqsubseteq)$ é um Domínio

Nessa seção

- definimos o conceito de *domínio*, e
- provamos que $(\Sigma_\perp \to \Sigma_\perp, \sqsubseteq)$ é um *domínio*

Esse fato será usado mais tarde em outra seção. Precisamos de uma série de outras definições antes de chegarmos a definição de domínio. Por motivos didáticos ilustraremos essas definições (incluindo a definição de domínio) utilizando também outros conjuntos e não somente o conjunto $\Sigma_\perp \to \Sigma_\perp$

Ordem Parcial (205)

Definição 1 Um relação binária $\sqsubseteq$ em um conjunto D é uma *ordem parcial* sse ela é

- reflexiva: $\forall d \in D.\ d \sqsubseteq d$
- transitiva: $\forall d, d', d'' \in D.\ d \sqsubseteq d' \sqsubseteq d'' \Rightarrow d \sqsubseteq d''$
- anti-simétrica: $\forall d, d', \in D.\ d \sqsubseteq d' \sqsubseteq d \Rightarrow d = d'$

Dizemos que $(D, \sqsubseteq)$ é um *conjunto parcialmente ordenado* ou um *poset*.

Exercício 1 **Prove que** $(\Sigma_\perp \to \Sigma_\perp, \sqsubseteq)$ **é uma ordem parcial.**

Exercício 2 Dê 2 exemplos de ordem parcial e 2 exemplos de ordens que não são parciais (os exemplos podem ser ordens "naturais", ou podem ser ordens criadas artificialmente.)

Menor Elemento (206)

Definição 2 Supor que D é um poset. Um elemento $d \in D$ é o menor elemento de D se ele satisfaz

$$\forall d' \in D.\ d \sqsubseteq d'.$$

Note que D tem no máximo um menor elemento pois $\sqsubseteq$ é anti-simétrica (prove esse fato)

- Nem todo poset possui menor elemento, $(\mathbb{Z}, \leq)$, por exemplo, não possui.
- Se existir, nos referimos ao menor elemento de um poset D como $\perp_D$ (ou somente $\perp$ quando D está claro pelo contexto). O elemento $\perp$ é, portanto, unicamente determidado pela seguinte propriedade $\forall d \in D.\ \perp \sqsubseteq d$. O menor elemento de um poset as vezes é chamado de *botton*.

Menor Elemento (207)

O menor elemento do poset $(\Sigma_\perp \to \Sigma_\perp, \sqsubseteq)$ é a função totalmente indefinida

$$\lambda\, \sigma \in \Sigma_\perp.\ \perp.$$

Exercício 3 Prove que $\lambda\,\sigma \in \Sigma_\perp.\ \perp$ satisfaz o requisito para ser o menor elemento de $(\Sigma_\perp \to \Sigma_\perp, \sqsubseteq)$

Cadeias e Limites (208)

- Uma *cadeia* ascendente contável em um poset D é uma seqüência de elementos de D

$$d_0 \sqsubseteq d_1 \sqsubseteq d_2 \sqsubseteq \dots$$

- Um *limite superior* para a cadeia é qualquer $d \in D$ satisfazendo $\forall n \in \mathbb{N}.\ d_n \sqsubseteq d$.
- Nem toda cadeia possui limite superior: cadeia $0 \leq 1 \leq 2 \leq 3 \dots$ no poset $(\mathbb{N}, \leq)$, por exemplo, não possui.

Cadeias e Limites (209)

- Cadeias de posets podem também ter mais do que um limite superior.
- Considere o conjunto $D = N \cup \{\omega_1, \omega_2\}$ equipado com a ordem definida como segue:

$$d \sqsubseteq d' \text{ sse } \begin{cases} & d, d' \in \mathbb{N}\ e\ d \leq d' \\ ou & d \in \mathbb{N}\ ed' \in \{\omega_1, \omega_2\} \\ ou & d = d' = \omega_1 \\ ou & d = d' = \omega_2 \end{cases}$$

- A cadeia $0 \sqsubseteq 1 \sqsubseteq 2 \sqsubseteq 3 \dots$ no poset acima, por exemplo possui 2 limites superiores (ω_1 e ω_2.

Cadeias e Limites (210)

- Se existir, o *menor limite superior* (*lub*) de uma cadeia $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \sqsubseteq \dots$ será representado por

$$\bigsqcup_{n=0}^{\infty} d_n.$$

- e deve satisfazer:
  - $\forall m \in \mathbb{N}.\ d_m \sqsubseteq \bigsqcup_{n=0}^{\infty} d_n$
  - $\forall d \in D.\ \forall m \in \mathbb{N}.\ \ d_m \sqsubseteq d\ \Rightarrow\ \bigsqcup_{n=0}^{\infty} d_n \sqsubseteq d.$

A cadeia $0 \sqsubseteq 1 \sqsubseteq 2 \sqsubseteq 3 \ldots$ no poset acima, possui 2 limites superiores (ω_1 e ω_2) mas nem um deles é menor do que o outro.
Não será necessário considerarmos cadeias incontáveis ou decrescentes em um poset, logo de agora em diante *cadeia* será usado para *cadeia contável ascendente.*

Cadeias e Limites (211)

- Os elementos de uma cadeia não precisam ser distintos. Em particular, dizemos que uma cadeia $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \sqsubseteq \ldots$ é *eventualmente constante* se, a partir de alguma posição $n \in \mathbb{N}$ da cadeia o elemento é sempre o mesmo.
- Note que, nesse caso, $\bigsqcup_{n=0}^{\infty} d_n$ é um elemento de que faz parte da cadeia
- Assim como o menor elemento de um poset, o lub de uma cadeia é único, se existir.
- O lub as vezes é chamado de *supremo*. Outras notações comuns para $\bigsqcup_{n=0}^{\infty} d_n$ são

$$\bigsqcup_{n\geq 0} d_n \quad e \quad \bigsqcup\{d_n \mid n \geq 0\}.$$

- Se tirarmos qualquer quantidade finita de elementos do começo de uma cadeia não mudamos o seu lub. Por exemplo:

$$\bigsqcup_{n=0}^{\infty} d_n \quad = \quad \bigsqcup_{n=1}^{\infty} d_n$$

Para semântica denotacional estamos interessados em posets com determinadas propriedades:

Ordem Parcial Completa e Domínio (212)

Definição 3 Uma *ordem parcial completa* (*cpo*) é um poset $(D, \sqsubseteq)$ no qual todas as cadeias contáveis ascendentes $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \ldots$ possuem lubs $\bigsqcup_{n=0}^{\infty} d_n$

Exercício 4 Prove que $(\Sigma_\perp \rightarrow \Sigma_\perp, \sqsubseteq)$ é uma ordem parcial completa.

Definição 4 Um **domínio** é um cpo com menor elemento $\perp$.

Exercício 5 Prove que $(\Sigma_\perp \rightarrow \Sigma_\perp, \sqsubseteq)$ é um domínio.

4.5.2 O funcional F que surge a partir de IMP é contínuo

Funções Contínuas (213)

- Uma função $f : D \rightarrow E$ entre posets $(D, \sqsubseteq_D)$ $(E, \sqsubseteq_E)$ é *monotônica* (ou seja preserva ordem) sse $\forall d, d' \in D.\ d \sqsubseteq_D d'$ implica $f(d) \sqsubseteq_E f(d')$

- Se D e E são também cpos, a função f é *contínua* sse é monotônica e preserva lubs de cadeias, ou seja , para todas cadeias $d_0 \sqsubseteq_D d_1 \sqsubseteq_D d_2 \sqsubseteq_D \ldots$ em D, é o caso que

$$f(\bigsqcup_{n=0}^{\infty} d_n) = \bigsqcup_{n=0}^{\infty} f(d_n)$$

Exercício 6 Prove que o funcional F da equação de ponto fixo associada ao comando `while` é contínuo.

Se D e E possuem menor elemento, então a função f é estrita sse $f(\bot) = \bot$
Observação: Note que se $f : D \to E$ é monotônica e $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \sqsubseteq \ldots$ é uma cadeia em D, então aplicando f obtemos a cadeia $f(d_0) \sqsubseteq f(d_1) \sqsubseteq f(d_2) \sqsubseteq \ldots$ em E. Além disso se d é um limite superior da primeira cadeia, então $f(d)$ é um limite superior da segunda cadeia e portanto é maior que o seu lub. Logo temos que se $f : D \to E$ é função monotônica entre cpos sempre temos que

$$\bigsqcup_{n=0}^{\infty} f(d_n) \sqsubseteq_E f(\bigsqcup_{n=0}^{\infty} d_n).$$

Consequentemente, usando a propriedade anti-simétrica de $\sqsubseteq$, para verificar se uma função parcial monotônica entre cpos é contínua, é suficiente verificar, para cada cadeia $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \sqsubseteq \ldots$ em D, se

$$f(\bigsqcup_{n=0}^{\infty} d_n) \sqsubseteq \bigsqcup_{n=0}^{\infty} f(d_n)$$

vale em E.

4.5.3 Teorema do Ponto Fixo

Teorema do Ponto Fixo (214)

O seguinte teorema afirma uma propriedade fundamental de domínios e funções contínuas importante para a semântica denotacional do comando while.

Teorema 1 Se D é um domínio e $f : D \to D$ uma função contínua então

$$x = \bigsqcup_{n \geq 0}^{\infty} f^n(\bot)$$

é o menor ponto fixo de f. Em outras palavras $fx = x$ e, sempre que $fy = y$, $x \sqsubseteq y$

PROVA. claramente $\bot \sqsubseteq f\bot$, e se $f^n\bot \sqsubseteq f^{n+1}\bot$, então $f^{n+1}\bot = f(f^n\bot) \sqsubseteq f(f^{n+1}\bot) = f^{n+2}\bot$ já que f é monotônica. Assim, por indução em n, $\bot \sqsubseteq f\bot \sqsubseteq f^2\bot \sqsubseteq f^3\bot \ldots$ é uma cadeia de talforma que x é bem definido.

Então, como f é contínua e x é lub de cadeia:

$$fx = f(\bigsqcup_{n\geq 0}^{\infty} f^n(\bot)) = \bigsqcup_{n\geq 0}^{\infty} f^{n+1}(\bot) = \bigsqcup_{n\geq 1}^{\infty} f^n(\bot) = \bigsqcup_{n\geq 0}^{\infty} f^n(\bot) = x$$

E por fim, suponha que $fy = y$. Então $\bot \sqsubseteq y$ e, se $f^n\bot \sqsubseteq y$ então $f^{n+1}\bot = f(f^n\bot) \sqsubseteq fy \sqsubseteq y$. Logo por indução em n, y é um limite superior para todo $f^n\bot$ logo $x = \bigsqcup_{n\geq 0}^{\infty} f^n(\bot) \sqsubseteq y$.

De Volta a Semântica do `while` (215)

- o conjunto $\Sigma_\bot \to \Sigma_\bot$ é um domínio, e
- o funcional F é contínuo

Logo, pelo Teorema do Ponto Fixo, podemos afirmar que o significado do comando while é dado pela seguinte equação semântica.

$$[\![\texttt{while}\ \ b\ c]\!] = \bigsqcup_{n\geq 0}^{\infty} F^n(\bot)$$

onde: $F \in (\Sigma_\bot \to \Sigma_\bot) \to (\Sigma_\bot \to \Sigma_\bot)$ *é definido como:*

$$Fg = \ \lambda\sigma \in \Sigma_\bot.\ se\ [\![b]\!]\sigma\ \mathit{então}\ g([\![c]\!]\sigma)\ \mathit{senão}\ \sigma$$