

FLP - examen 2024

FMI, Info, Anul II
Semestrul II, 2023/2024
Fundamentele limbajelor de programare

Examen

1. Fie $I, N, P \in L$, distincte două câte două, și $K \in V$. Notăm

$\text{while } I * I \leq N \text{ do } ((\text{if } I * I = N \text{ then } P := 1 \text{ else skip}); I := I + 1))$

cu **Pgm**.

- (a) (2 puncte) Să se descrie formal execuția lui **Pgm**, dintr-o stare inițială σ cu $\sigma(N) = 30$, $\sigma(I) = 5$, $\sigma(P) = 0$, folosind semantica operațională big-step SAU cea small-step.
- (b) (2 puncte) Să se arate că enunțul Hoare

$$\{I = 0 \wedge \exists K(N = K * K)\} \text{Pgm} \{P = 1\}$$

este demonstrabil.

1. a) while $I \neq N$ do ((if $I \neq N$ then $P := 1$ else skip); $I := I + 1$))

$$\begin{aligned} \bar{v}(N) &= 30 \\ \bar{v}(I) &= 5 \\ \bar{v}(P) &= 0 \end{aligned}$$

w = while....
i = if...

Big step

$$\frac{e_{\bar{v}}(I \neq N) = 1 \quad (i; I := I + 1, \bar{v}) \Downarrow \bar{v}_2 \quad (w, \bar{v}_2) \Downarrow \bar{v}_1}{(w, \bar{v}) \Downarrow \bar{v}_1}$$

$$\frac{(i, \bar{v}) \Downarrow \bar{v}_3 \quad (I := I + 1, \bar{v}_3) \Downarrow \bar{v}_2}{(i; I := I + 1, \bar{v}) \Downarrow \bar{v}_2}$$

$$\Rightarrow \bar{v}_3 = \bar{v}$$

$$\frac{e_{\bar{v}}(I \neq N) = 0, (\text{skip}, \bar{v}) \Downarrow \bar{v}}{(i, \bar{v}) \Downarrow \bar{v}}$$

$$\frac{(I := I + 1, \bar{v}_3) \Downarrow \bar{v}_3}{I \rightarrow e_{\bar{v}_3}(I + 1) = \bar{v}_2}$$

$$\begin{aligned} \bar{v}_2(N) &= 30 \\ \bar{v}_2(I) &= 6 \\ \bar{v}_2(P) &= 0 \end{aligned}$$

$$\frac{e_{\bar{v}_2}(I \neq N) = 0}{(w, \bar{v}_2) \Downarrow \bar{v}_2}$$

$$\Rightarrow \bar{v}_1 = \bar{v}_2 = \text{state final} \quad \square$$

Small step

$$(w, \sigma) \rightarrow (i; \underbrace{I * I \leq N}_{i'} \text{ then } (i; I := I + 1); w) \text{ else skip}, \sigma)$$

$$\frac{e_{\sigma}(I * I \leq N) = 1}{(i', \sigma) \rightarrow ((i; I := I + 1); w, \sigma)}$$

$$\frac{e_{\sigma}(I * I = N) = 0}{(i, \sigma) \rightarrow (\text{skip}, \sigma)}$$

$$\frac{(i, \sigma) \rightarrow (\text{skip}, \sigma)}{(i; I := I + 1, \sigma) \rightarrow (\text{skip}; I := I + 1, \sigma)}$$

$$\downarrow$$

$$(I := I + 1, \sigma) \rightarrow (\text{skip}, \sigma_{I-1}, e_{\sigma}(I + 1))$$

$$\downarrow$$

$$\sigma_1$$

$$\begin{aligned} \sigma_1(N) &= 30 \\ \sigma_1(I) &= 6 \\ \sigma_1(P) &= 0 \end{aligned}$$

$$\frac{(i; I := I + 1, \sigma) \rightarrow (\text{skip}, \sigma_1)}{((i; I := I + 1); w, \sigma) \rightarrow (\text{skip}; w, \sigma_1)}$$

$$\downarrow$$

$$(w, \sigma_1)$$

$$(w, \sigma_1) \rightarrow (i', \sigma_1)$$

$$\frac{e_{\sigma_1}(I * I \leq N) = 0}{(i', \sigma_1) \rightarrow (\text{skip}, \sigma_1) \quad \square}$$

$$b) \{ I = 0 \wedge \exists k (N = k * k) \} Pgm \{ P = 1 \}$$

$$\{ A \} w \{ A \wedge \neg (I * I \leq N) \} \models \{ I = 0 \wedge \exists k (N = k * k) \} \neg \{ A \} \\ \models \{ A \wedge \neg (I * I \leq N) \} \rightarrow \{ P = 1 \}$$

$$\{ A \wedge (I * I \leq N) \{ i; I := I + 1 \} \{ A \}$$

$$\Rightarrow \{ A \wedge (I * I \leq N) \{ i; B \} \Rightarrow \{ A \wedge (I * I \leq N) \wedge (I * I = N) \} \{ P := 1 \} \{ B \} \quad (a) \\ \{ B \} \{ I := I + 1 \} \{ A \} \quad \{ A \wedge (I * I \leq N) \wedge \neg (I * I = N) \} \{ skip \} \{ B \} \quad (b)$$

$$B = A \{ I := I + 1 \}$$

(a)

$$\models \{ A \wedge (I * I \leq N) \wedge (I * I = N) \} \rightarrow \{ B \{ P := 1 \} \} \\ \models \{ B \} \rightarrow \{ P = 1 \}$$

(b)

$$\models \{ A \wedge (I * I \leq N) \wedge \neg (I * I = N) \} \rightarrow \{ B \}$$

$$A = \{ (I - 1) * (I - 1) = N \}$$

2. (2 puncte) Considerăm o semnătură de ordinul 1 în care avem simbolurile de funcție f, g, h cu aritățile 2, 1 și 3, respectiv. Fie x, y variabile. Aplicați algoritmul de unificare din curs pentru mulțimea de ecuații

$$\{ h(x, y, f(g(x), g(y))) = h(y, x, f(y, x)) \}.$$

Explicitați aplicarea fiecărui pas, menționând pasul, ecuația folosită și mulțimea nouă de ecuații obținută după aplicarea pasului.

$$2. \{ h(x, y, f(g(x), g(y))) = h(y, x, f(y, x)) \}$$

$$\{ x \neq y, y = x, f(g(x), g(y)) = f(y, x) \}$$

$$\Rightarrow \{ y \neq x, f(g(y), g(y)) = f(y, y) \}$$

$$\Rightarrow \{ g(y) \neq y, g(y) = y \}$$

$$\Rightarrow \{ y = g(y), y = g(y) \}$$

$$\Rightarrow \text{"Esec" } (y \text{ este în mulțimea var. lui } g(y))$$

3. (2 puncte) Găsiți o SLD-respingere pentru următorul program Prolog

`shuffle(lit(X), lit(X)).`

`shuffle(arb(X, Y, Z), arb(T, U, W)) :- shuffle(X, U), shuffle(Y, W), shuffle(Z, T).`

și ținta:

`shuffle(X, arb(arb(lit(t), lit(c), lit(u)), arb(lit(i), lit(a), lit(t)), arb(lit(e), lit(t), lit(r))))`

În plus, precizați valoarea lui X în substituția calculată.

3.
$$\text{shuffle}(X, \underbrace{\text{arb}(\text{arb}(\text{lit}(t), \text{lit}(c), \text{lit}(u)), \text{lit}(i), \text{lit}(a), \text{lit}(t))}_{\text{arb}(A, B, C)}, \underbrace{\text{arb}(\text{lit}(e), \text{lit}(t), \text{lit}(r))}_{\text{lit}(e)})$$

▷
$$\begin{aligned} &\text{shuffle}(A, \text{arb}(\text{lit}(i), \text{lit}(a), \text{lit}(t))), \\ &\text{shuffle}(B, \text{arb}(\text{lit}(c), \text{lit}(t), \text{lit}(r))), \\ &\text{shuffle}(C, \text{arb}(\text{lit}(t), \text{lit}(c), \text{lit}(u))) \end{aligned}$$

$A = \text{arb}(A_1, A_2, A_3)$
 $B = \dots$
 $C = \dots$

▷
$$\begin{aligned} &\text{shuffle}(A, \text{lit}(a)), \text{shuffle}(A_2, \text{lit}(t)), \text{shuffle}(A_3, \text{lit}(i)), \\ &\text{shuffle}(B_1, \text{lit}(t)), \dots \end{aligned}$$

▷ □

$A_1 = \text{lit}(a), A_2 = \text{lit}(t), \dots$

$$x = \text{arb}(A, B, C) = \text{arb}(\text{arb}(\text{lit}(a), \text{lit}(t), \text{lit}(i)), \text{arb}(\text{lit}(t), \text{lit}(r), \text{lit}(c)), \text{arb}(\text{lit}(c), \text{lit}(u), \text{lit}(t)))$$

4. (2 puncte) Fie λ -termenul

$$t := \lambda r. (\lambda e. ((\lambda d. d)(er))).$$

Să se găsească τ și o demonstrație că

$$\vdash t : \tau.$$

$$\begin{aligned}
4. \quad & t := \lambda r. (\lambda e. ((\lambda d. d_1(er))) \\
& M = \lambda r. R. (\lambda e. E. (\lambda d. D. d_1(er))) \\
& r_M := \{x : X \mid x \in FV(M)\} \\
& cc(M, r_M, A) = cc(M_2, r_{M_1} = r_M \cup \{r : R\}, A_1) \cup \{A = R \rightarrow A_1\} = \\
& = cc(M_2, r_{M_2} = r_{M_1} \cup \{e : E\}, A_2) \cup \underbrace{k_1 \cup \{A_1 = E \rightarrow A_2\}}_{k_2} = \\
& = cc(\lambda d. D. d, r_{M_2}, B_1) \cup cc(er, r_{M_2}, B_2) \cup \underbrace{k_2 \cup \{B_1 = B_2 \rightarrow A_2\}}_{k_3} = \\
& = cc(d, r_{M_2} \cup \{d : D\}, C_1) \cup \underbrace{\{B_1 = D \rightarrow C_1\}}_{k_4} \\
& \cup cc(e, r_{M_2}, C_2) \cup cc(r, r_{M_2}, C_3) \cup \underbrace{\{C_2 = C_3 \rightarrow B_2\}}_{k_5} \cup k_3 = \\
& = \{D = C_1\} \cup \{E = C_2\} \cup \{R = C_3\} \\
& \cup \{A = R \rightarrow A_1\} \cup \{A_1 = E \rightarrow A_2\} \cup \{B_1 = B_2 \rightarrow A_2\} \cup \{B_1 = D \rightarrow C_1\} \\
& \cup \{C_2 = C_3 \rightarrow B_2\} \\
& = \{A = C_3 \rightarrow A_1\} \cup \{A_1 = C_2 \rightarrow A_2\} \cup \{B_1 = B_2 \rightarrow A_2\} \\
& \cup \{B_1 = C_1 \rightarrow C_1\} \cup \{C_2 = C_3 \rightarrow B_2\} \\
& \quad C_1 = B_2 = A_2 \\
& \Rightarrow \{A = C_3 \rightarrow (C_2 \rightarrow C_1)\} \cup \{B_1 = C_1 \rightarrow C_1\} \cup \{C_2 = C_3 \rightarrow C_1\} = \\
& = \{A = C_3 \rightarrow (C_3 \rightarrow C_1) \rightarrow C_1\} \cup \{B_1 = C_1 \rightarrow C_1\}
\end{aligned}$$

5. (2 puncte) Fie b o expresie booleană și c o instrucțiune. Să se arate că, pentru orice $(\sigma, \sigma') \in \Sigma^2$, $(\sigma, \sigma') \in \llbracket \text{while } b \text{ do } c \rrbracket$ dacă și numai dacă există $n \geq 0$ și un șir finit de stări $(\sigma_i)_{i \leq n}$ cu $\sigma_0 = \sigma$, $\sigma_n = \sigma'$, $\llbracket b \rrbracket(\sigma_n) = 0$ și, pentru orice i cu $0 \leq i < n$, $\llbracket b \rrbracket(\sigma_i) = 1$ și $(\sigma_i, \sigma_{i+1}) \in \llbracket c \rrbracket$.

demonstratie cursul 3 (?)

6. (bonus: 1 punct) Descrieți punctual, dacă există, un moment din curs care v-a schimbat modul cum priviți activitatea de a programa.

Din acest curs despre Fundamentele Limbajelor de Programare, un concept care poate schimba modul în care privești programarea este legat de tipurile de date abstracte și inductive.

Acesta este un moment revelator deoarece:

- Înțelegi că datele pe care le manipulezi în programare nu sunt doar simple valori, ci pot fi definite inductiv, într-un mod matematic riguros
- Vezi cum formulele și expresiile pe care le folosești în programare sunt de fapt construite din reguli precise de formare, similar cu modul în care se construiesc formulele în logică
- Realizezi că există o fundamentare matematică solidă în spatele constructelor pe care le folosești în limbaje precum Haskell, unde definești tipuri de date cu constructori

Această perspectivă îți poate schimba radical modul în care privești programarea, trecând de la o abordare pur practică la înțelegerea fundamentelor teoretice care stau la baza limbajelor de programare.