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, şi $K \in V$. Notăm

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

cu **Pgm**.

(a) (2 puncte) Să se descrie formal execuţia lui **Pgm**, dintr-o stare iniţială σ cu $\sigma(N) = 30$, $\sigma(I) = 5$, $\sigma(P) = 0$, folosind semantica operaţională big-step SAU cea small-step.

(b) (2 puncte) Să se arate că enunţul Hoare

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

este demonstrabil.

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

$\sigma(N)=30$
$\sigma(I)=5$
$\sigma(P)=0$

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

Big step

$$\frac{e_\sigma(I*I\leq N)=1 \quad (i;I:=I+1,\sigma)\Downarrow\sigma_2 \quad (w,\sigma_2)\Downarrow\sigma_1}{(w,\sigma)\Downarrow\sigma_1}$$

$$\frac{(i,\sigma)\Downarrow\sigma_3 \quad (I:=I+1,\sigma_3)\Downarrow\sigma_2}{(i;I:=I+1,\sigma)\Downarrow\sigma_2}$$

$\Rightarrow \sigma_3=\sigma$

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

$$\frac{}{(I:=I+1,\sigma_3)\Downarrow\sigma_3}\ I\to e_{\sigma_3}(I+1) \quad =\sigma_2$$

$\sigma_2(N)=30$
$\sigma_2(I)=6$
$\sigma_2(P)=0$

$$\frac{e_{\sigma_2}(I*I\leq N)=0}{(w,\sigma_2)\Downarrow\sigma_2}$$

$\Rightarrow \sigma_1=\sigma_2$ = stare finală □

Small step

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

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

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

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

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

$\sigma_{I \to e_\sigma(I+1)} = \sigma_1$

$\sigma_1(N) = 30$
$\sigma_1(I) = 6$
$\sigma_1(P) = 0$

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

$$(w, \sigma_1)$$

$$\frac{}{(w, \sigma_1) \to (i', \sigma_1)}$$

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

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

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

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

$\{A \wedge (I*I \leq N)\}$ (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) $\{B\}$ (a)

$\{B\}$ (I := I+1) $\{A\}$ $\{A \wedge (I*I \leq N) \wedge \neg(I*I = N)\}$ (skip) $\{B\}$ (b)

$B = A[I := I+1]$

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

(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 signatură de ordinul I în care avem simbolurile de funcţie f, g, h cu arităţile 2, 1 şi 3, respectiv. Fie x, y variabile. Aplicaţi algoritmul de unificare din curs pentru mulţimea de ecuaţii

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

Explicitaţi aplicarea fiecărui pas, menţionând pasul, ecuaţia folosită şi mulţimea nouă de ecuaţii obţinută după aplicarea pasului.

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

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

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

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

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

$\Rightarrow$ "Esec" (y este in multimea vor. lui g(y))

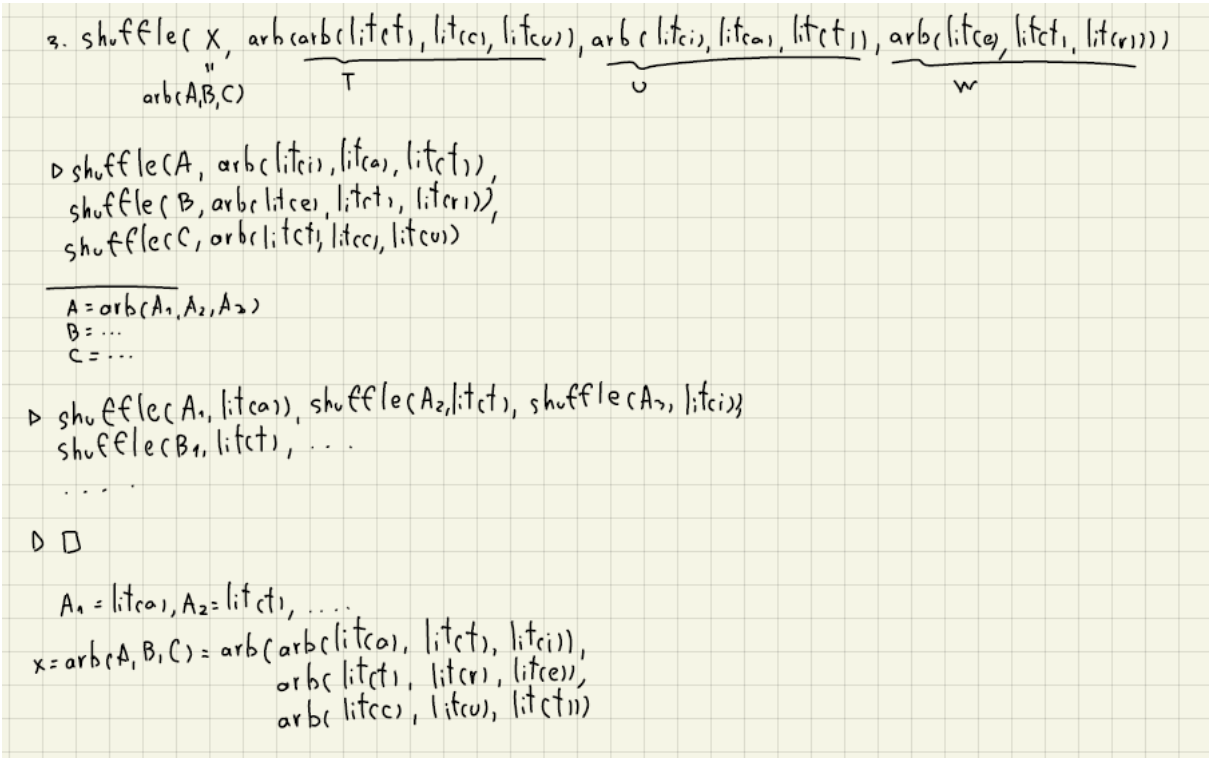
3. (2 puncte) Găsiţi o SLD-respingere pentru următorul program Prolog

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

şi ţinta:

```
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, precizaţi valoarea lui X în substituţia calculată.

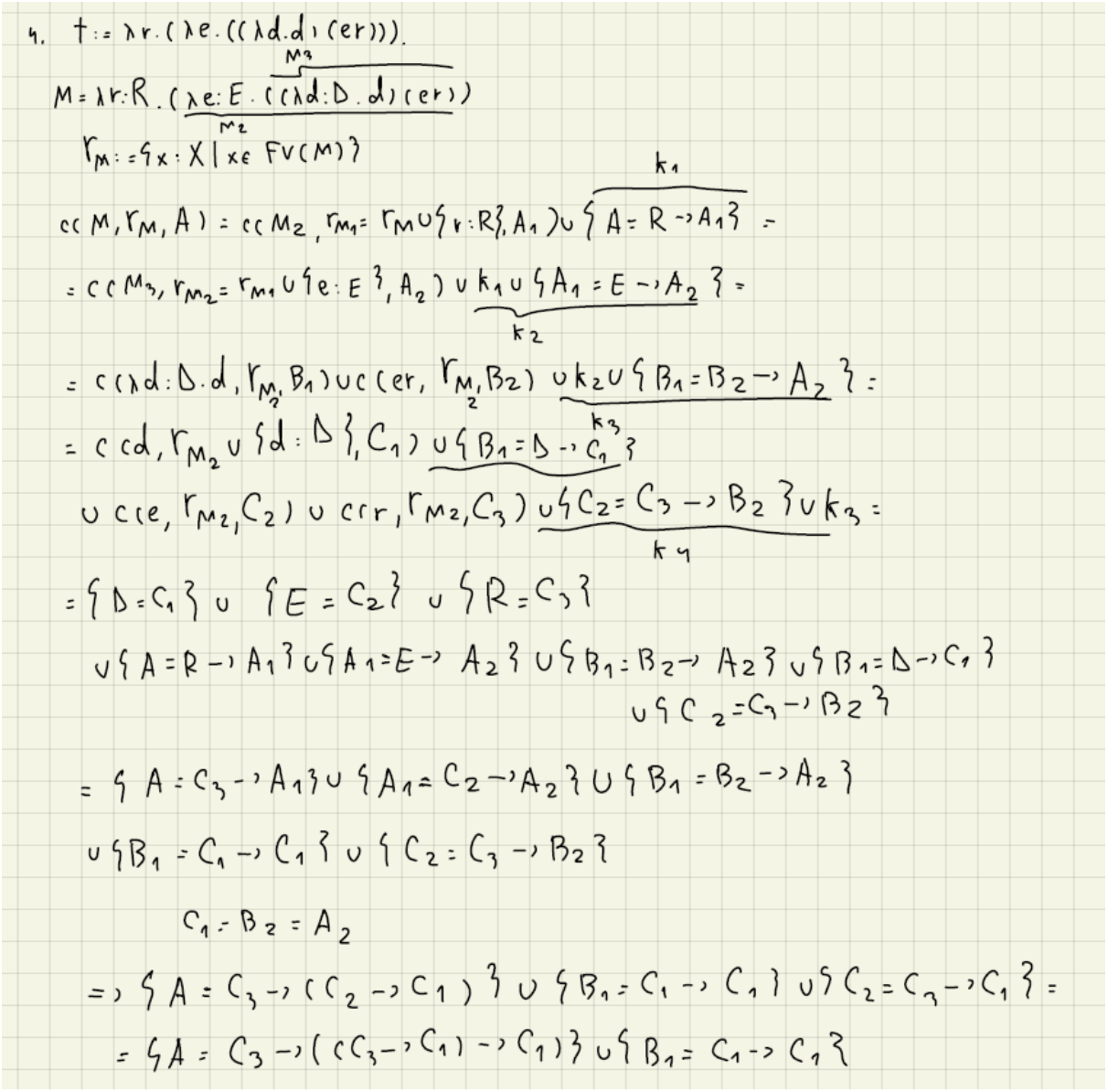


4. (2 puncte) Fie λ-termenul

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

Să se găsească τ şi o demonstraţie că

$$\vdash t : \tau.$$



5. **(2 puncte)** Fie b o expresie booleană şi c o instrucţiune. Să se arate că, pentru orice $(\sigma, \sigma') \in \Sigma^2$, $(\sigma, \sigma') \in [\![\textbf{while } b \textbf{ do } c]\!]$ dacă şi numai dacă există $n \geq 0$ şi un şir finit de stări $(\sigma_i)_{i \leq n}$ cu $\sigma_0 = \sigma$, $\sigma_n = \sigma'$, $[\![b]\!](\sigma_n) = 0$ şi, pentru orice i cu $0 \leq i < n$, $[\![b]\!](\sigma_i) = 1$ şi $(\sigma_i, \sigma_{i+1}) \in [\![c]\!]$.

demonstratie cursul 3 (?)

6. **(bonus: 1 punct)** Descrieţi punctual, dacă există, un moment din curs care v-a schimbat modul cum priviţi 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.