BEVEZETÉS A PROGRAMOZÁSHOZ

(Kivonat)

ELSŐ RÉSZ

A jegyzet elsősorban E.W. Dijkstra alapvető munkáira támaszkodik, az általa kimondott elveket igyekszik követni. Célja egy tudományos igényű programozói szemlélet kialakítása. $\boxed{1}$

FELHASZNÁLT MATEMATIKAI FOGALMAK

1. Sorozatok

Jelölések

$\mathcal{N}$ a természetes számok halmaza.

$\mathcal{N}_0$ a természetes számokból és a nullából álló halmaz.

$\mathcal{Z}$ az egész számok halmaza.

$\mathcal{L} \overset{\text{def}}{=} \{hamis, igaz\}$, a logikai értékek halmaza.

Véges sorozat: $\alpha = (\alpha_1, \dots, \alpha_n)$.

Végtelen sorozat: $\alpha = (\alpha_1, \dots\dots)$.

A^* : A elemeiből képzett véges sorozatok halmaza.

A^∞ : A elemeiből képzett végtelen sorozatok halmaza.

$A^{**} \overset{\text{def}}{=} A^* \cup A^\infty$.

Definíciók

1.) $kon(\alpha, \beta)$ az $\alpha \in A^*$ és a $\beta \in A^{**}$ sorozatok ***konkatenált***ja, ha
$kon(\alpha, \beta) \overset{\text{def}}{=} (\alpha_1, \dots, \alpha_n, \beta_1 \dots\dots)$.

2.) Az $\alpha \in A^{**}$ ***redukált***jának nevezzük, és $red(\alpha)$-val jelöljük azt a sorozatot, amit úgy kapunk, hogy az α minden azonos elemekből álló véges részsorozatát a részsorozat egyetlen elemével helyettesítjük.

3.) Legyen $\alpha^1, \alpha^2, \dots, \alpha^{n-1} \in A^*$; $\alpha^n \in A^{**}$.
$\chi_n(\alpha^1, \alpha^2, \dots, \alpha^n) \overset{\text{def}}{=} red(kon(\alpha^1, \alpha^2, \dots, \alpha^n))$.
χ_∞ : végtelen sok véges sorozat konkatenáltjának redukáltja.

4.) Legyen $\alpha = (\alpha_1, \dots, \alpha_n) \in A^*$. $\tau : A^* \to A$ olyan függvény, amelyre $\tau(\alpha) \overset{\text{def}}{=} \alpha_n$.

2. Direktszorzat

Definíciók

1.) Legyen $I = \{i_1, i_2, \dots, i_n\}$. $\quad \underset{i_j \in I}{\times} A_{i_j} \overset{\text{def}}{=} \{(a_{i_1}, a_{i_2}, \dots, a_{i_n}) \mid a_{i_j} \in A_{i_j}\}$.

2.) Azt mondjuk, hogy a B halmaz az A halmaz ***altere***, ha $A = \underset{i_j \in I}{\times} A_{i_j}$; $B = \underset{j_k \in J}{\times} A_{j_k}$, ahol $I = \{i_1, i_2, \ldots, i_n\}$, $J = \{j_1, j_2, \ldots, j_m\}$, $m \leq n$ és $J \subseteq I \subseteq \mathcal{N}$.

3.) $pr_B\,(a)$ az $a \in A$-nak a B-re vett *projekció*ja, ha B altere A-nak és $pr_B\,(a) \overset{\text{def}}{=} (a_{j_1}, a_{j_2}, \ldots, a_{j_m})$ (a 2. pont jelöléseivel).

3. Relációk

Definíciók

1.) Az $R \subseteq A \times B$-t *bináris relációnak* nevezzük.

2.) Legyen $R \subseteq A \times B$. $R(a) \overset{\text{def}}{=} \{\, b \,|\, (a,b) \in R\,\}$.

3.) Legyen $R \subseteq A \times B$. Az R *reláció értelmezési tartománya*: $\mathcal{D}_R \overset{\text{def}}{=} \{\, a \in A \,|\, \exists b \in B : (a,b) \in R\}$.

4.) Legyen $R \subseteq A \times B$. Az R *reláció értékkészlete*: $\mathcal{R}_R \overset{\text{def}}{=} \bigcup_{a \in A} R(a)$.

5.) Az $R \subseteq A \times B$ reláció *determinisztikus reláció* (vagy parciális függvény), ha $\forall\, a \in A \,:\, |\,R(a)\,| \leq 1$.

6.) Az $R \subseteq A \times B$ reláció *függvény*, ha $\forall\, a \in A \,:\, |\,R(a)\,| = 1$.

7.) $R^{(-1)} \overset{\text{def}}{=} \{\,(b,a) \,|\, (a,b) \in R\,\}$ az $R \subseteq A \times B$ *reláció inverze.*

8.) $\widetilde{R} \overset{\text{def}}{=} \{\,(a,b) \,|\, (a,b) \notin R \wedge (a,b) \in A \times B\}$ az $R \subseteq A \times B$ *reláció komplemense.*

9.) $R_2 \circ R_1 \overset{\text{def}}{=} \{\,(a,c) \,|\, \exists\, b \in B \,:\, (a,b) \in R_1 \wedge (b,c) \in R_2\,\}$ az $R_1 \subseteq A \times B$ és az $R_2 \subseteq B \times C$ relációk *kompozíció*ja.

10.) $R^n \overset{\text{def}}{=} R \circ R^{n-1}$ $(n \in \mathcal{N})$ az $R \subseteq A \times A$ reláció n. *hatvány*a.

11.) Legyen $R \subseteq A \times A$. $R^0 \overset{\text{def}}{=} \{\,(a,a) \,|\, a \in A\}$.

12.) $R_2 \odot R_1 \overset{\text{def}}{=} \{\,(a,c) \,|\, R_1\,(a) \subseteq \mathcal{D}_{R_2} \wedge \exists\, b \in B \,:\, (a,b) \in R_1 \wedge (b,c) \in R_2\,\}$ az $R_1 \subseteq A \times B$ és az $R_2 \subseteq B \times C$ relációk *szigorú kompozíció*ja.

13.) $R^{(-1)}\,(H) \overset{\text{def}}{=} \{a \in A \,|\, \exists\, h \in H \,:\, (a,h) \in R\}$ a $H \subseteq B$ halmaznak az $R \subseteq A \times B$ relációra vonatkozó *inverz kép*e.

14.) $R^{-1}\,(H) \overset{\text{def}}{=} \{a \in A \,|\, R\,(a) \subseteq H \wedge R\,(a) \neq \emptyset\}$ a $H \subseteq B$ halmaznak az $R \subseteq A \times B$ relációra vonatkozó *őskép*e.

15.) $L_R(a_0, \ldots, a_q)$ az $R \subseteq A \times A$ reláció q *hosszúságú pontlánc*a, ha $\forall i \in [1,q] : a_i \in R(a_{i-1})$. a_q a sorozat *végpont*ja, ha $a_q \notin \mathcal{D}_R$.

16.) $L_R(a_0, \ldots\ldots)$ az $R \subseteq A \times A$ reláció *végtelen pontlánc*a, ha $\forall i \in N : a_i \in R(a_{i-1})$.

17.) $\overline{R}$ az $R \subseteq A \times A$ reláció *lezárt*ja, ha $\mathcal{D}_{\overline{R}} \overset{\text{def}}{=} \{a \in A |\; \nexists L_R(a_0, \ldots\ldots) \,:\, a_0 = a\}$, $\overline{R}(a) \overset{\text{def}}{=} \{a' \in A | \exists L_R(a_0, \ldots, a_q) \,:\, a = a_0 \wedge a' = a_q \wedge a_q \notin \mathcal{D}_R\}$.

18.) $\overline{\overline{R}}$ az $R \subseteq A \times A$ reláció *korlátos lezárt*ja, ha

$$\mathcal{D}_{\overline{\overline{R}}} \overset{\text{def}}{=} \{a \in A |\; \nexists L_R(a_0, \ldots) : a_0 = a \wedge \exists n \in N : \forall L_R(a_0, \ldots, a_q) : (a = a_0) \to q \leq n\},$$

$$\overline{\overline{R}}(a) \overset{\text{def}}{=} \{a' \in A | \exists L_R(a_0, \ldots, a_q) \,:\, a = a_0 \wedge a' = a_q \,:\, a_q \notin \mathcal{D}_R\}.$$

19.) Az $f \subseteq A \times \mathcal{L}$ relációt *logikai függvény*nek (vagy állításnak) nevezzük.

20.) $[f] \stackrel{\text{def}}{=} \{a \in A | f(a) = \{igaz\}\}$ az f *logikai függvény igazsághalmaza.*

21.) $[R] \stackrel{\text{def}}{=} R^{-1}(\{igaz\})$ az $R \subseteq A \times \mathcal{L}$ *reláció igazsághalmaza.*

22.) Legyen $R \subseteq A \times B$, $H \subseteq A$.

$R|_H \stackrel{\text{def}}{=} (R \cap (H \times B))$ reláció az R megszorítása a H halmazra.

23.) Legyen $R \subseteq A \times A$ és π egy állítás A-n.

$R|_\pi \stackrel{\text{def}}{=} (R \cap ([\pi] \times A)) \cup \{(a, a) \mid [\pi] \setminus \mathcal{D}_R\}$ (*feltételre vonatkozó reláció*).

Megj.: $R|_\pi \neq R|_{[\pi]}$.

24.) $\overline{R|_\pi}$ az R reláció π feltételre vonatkozó lezártja, $\overline{\overline{R|_\pi}}$ az R reláció π feltételre vonatkozó korlátos lezártja.

Tételek

1.) $[f] = f^{-1}(\{i\})$.

2.) $\forall n \in N \, : \, |A| > n \Rightarrow \exists R \, : \, R \subseteq A \times A \, : \, \overline{R} \neq \overline{\overline{R}}$.

ALAPVETŐ DEFINÍCIÓK ÉS TÉTELEK

Definíciók

1.) Legyen $I \subset \mathcal{N}$, $\forall i_j \in I : A_{i_j}$ megszámlálható halmaz.

Az $A \stackrel{\text{def}}{=} \underset{i_j \in I}{\times} A_{i_j}$ halmazt *állapottér*nek, az A_{i_j} halmazokat *típusértékhalmaz*oknak nevezzük. $\boxed{2}$

2.) Az állapottér elemeit, az $a = (a_{i_1}, \ldots, a_{i_n}) \in A$ pontokat *állapot*oknak nevezzük.

3.) Az $a_{i_j} : A \to A_{i_j}$ projekciókat *változó*knak nevezzük. $\boxed{3}$

4.) Az $F \subseteq A \times A$ relációt *feladat*nak nevezzük. $\boxed{4}$

5.) Az $S \subseteq A \times A^{**}$ relációt *program*nak nevezzük, ha $\boxed{5}$

a.) $\forall a \in A \, : \, \forall \alpha \in S(a) \, : \, \alpha_1 = a$,

b.) $\mathcal{D}_S = A$,

c.) $\forall \alpha \in \mathcal{R}_S \, : \, \alpha = red(\alpha)$.

6.) A $p(S) \subseteq A \times A$ reláció az $S \subseteq A \times A^{**}$ program *programfüggvény*e, ha $\boxed{6}$

a.) $\mathcal{D}_{p(S)} = \{a \in A | S(a) \subseteq A^*\}$,

b.)$p(S)(a) = \{b \in A | \exists \alpha \in S(a) \, : \, \tau(\alpha) = b\}$.

7.) Az $S \subseteq A \times A^{**}$ program *megoldja* az $F \subseteq A \times A$ feladatot, ha $\boxed{7}$

a.) $\mathcal{D}_F \subseteq \mathcal{D}_{p(S)}$,

b.) $\forall a \in \mathcal{D}_F \, : \, p(S)(a) \subseteq F(a)$.

8.) Legyenek $S_1, S_2 \subseteq A \times A^{**}$ programok. S_1 *ekvivalens* S_2-vel, ha $p(S_1) = p(S_2)$.

9.) Legyen B altere az A állapottérnek. Az $F' \subseteq A \times A$ feladat az $F \subseteq B \times B$ *feladat kiterjesztése* az A térre, ha $F' = \{(a_1, a_2) \mid (pr_B(a_1), pr_B(a_2)) \in F\}$.

10.) Legyen B altere A-nak, $J \subseteq I$ és $A = \underset{i \in I}{\times} A_i$, $B = \underset{j \in J}{\times} A_j$. B' a B altér *kiegészítő* *altere* az A-ra, ha $B' = \underset{k\in(I\setminus J)}{\times} A_k$.

11.) Legyen B altere A-nak, B' a B altér kiegészítő altere az A-ra. S' az $S \subseteq B \times B^{**}$ *program kiterjesztése* A-ra, ha

$$\begin{aligned} S' = \{(a,\alpha) \mid\ & (pr_B(a), pr_B(\alpha)) \in S \wedge \\ & \alpha \in A^* \to \forall i \in [1..|\alpha|] : pr_{B'}(\alpha)_i = pr_{B'}(a) \wedge \\ & \alpha \in A^\infty \to \forall i \in \mathcal{N} : pr_{B'}(\alpha)_i = pr_{B'}(a)\} \end{aligned}$$

12.) Legyen B közös altere A_1-nek és A_2-nek, S_1 program A_1-en, S_2 program A_2-n. S_1 *ekvivalens* S_2-vel a B altéren, ha $pr_B(p(S_1)) = pr_B(p(S_2))$.

13.) Legyen B altere A-nak, $G \subseteq A \times A$ feladat. A G feladat ***vetítéstartó*** a B altér felett,ha

$\forall\ a_1, a_2 \in \mathcal{D}_G\ :\ (pr_B(a_1) = pr_B(a_2)\ \Rightarrow\ pr_B(G(a_1)) = pr_B(G(a_2)))$.

14.) Legyen B' altere A-nak, $G \subseteq A \times A$ feladat. A G feladat ***bővített identitás*** a B' altér felett, ha

$\forall\ (a_1, a_3) \in G :\ \exists (a_1, a_4) \in G : (pr_{B'}(a_4) = pr_{B'}(a_1)\ \wedge\ pr_B(a_3) = pr_B(a_4))$.

Tételek

Legyen B altere A-nak, B' a B kiegészítő altere A-ra, S program B-n, $F \subseteq B \times B$ feladat, S' illetve F' S-nek, illetve F-nek kiterjesztése A-ra, S_1 program A-n, $G \subseteq A \times A$ feladat, $G_1 = pr_{B\times B}(G)$, és legyen S_1 és S ekvivalens B-n. $\boxed{8}$

1.) Ha S' megoldja G-t $\Rightarrow$ S megoldja G_1-t.

2.) Ha S_1 megoldja F'-t $\Rightarrow$ S megoldja F-t.

3.) Ha S' megoldja F'-t $\Leftrightarrow$ S megoldja F-t.

4.) Ha S megoldja G_1-t, G vetítéstartó B felett, és bővített identitás B' felett $\Rightarrow$ S' megoldja G-t.

TÍPUS

Definíciók

1.) A $\mathcal{T}_S = (T, I_S, \mathcal{F})$ rendszert *típusspecifikáció*nak nevezzük.

T : alaphalmaz ;

I_S : *specifikációs invariáns* ; T_S ($= [I_S]$) : *típusértékhalmaz* ;

$\mathcal{F} = \{F_1, F_2, \ldots, F_n\}\ :\ \forall k \in [1,\ldots,n] : F_k \subseteq A_k \times A_k$;

$A_k = \underset{l\in L_k}{\times} A_{k\,l}\ \wedge\ \exists l \in L_k\ :\ A_{k\,l} = T_S$.

2.) $\mathcal{T} = (\varrho, \mathcal{S}, I)$ rendszert ***típus***nak nevezzük.

E : elemi típusértékek halmaza ;

ϱ : *reprezentációs függvény* ; $\varrho \subseteq E^* \times T'$;

$\mathcal{S}$: programok halmaza ; $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}\ :\ \forall i \in [1,\ldots,m] : S_i \subseteq B_i \times B_i$;

$B_i = \underset{j\in J_i}{\times} B_{ij}\ \wedge\ \exists j \in J_i\ :\ B_{ij} = E^*$.

I : ***típusinvariáns***, állítás az E^* halmazon.

3.) A $\mathcal{T} = (\varrho, \mathcal{S}, I)$ típus ***megfelel*** a $\mathcal{T}_S = (T, I_S, \mathcal{F})$ típusspecifikációnak, ha

a.) $T \subseteq T' \;\wedge\; \varrho([I]) = T_S$,

b.) $\forall F_k \in \mathcal{F} \;:\; (\exists S_i \in \mathcal{S} : S_i$ megoldja* F_k-t).

4.) $S_i \subseteq B_i \times B_i$ megoldja* $F_k \subseteq A_k \times A_k$-t, ha

a.) $B_i = \underset{j \in J_i}{\times} B_{ij} \;\wedge\; A_k = \underset{l \in L_k}{\times} A_{k\,l} \;\wedge\; L_k \subseteq J_i \subset \mathcal{N} \;\wedge$
$(\forall l \in L_k : B_{il} = A_{k\,l} \;\vee\; (B_{il} = E^* \;\wedge\; A_{k\,l} = T_S))$,

b.) $\forall a \in \mathcal{D}_{F_k} \quad : \quad \gamma_k \odot p(S) \odot \gamma_k^{(-1)}(a) \;\neq \emptyset$,

c.) $\forall a \in \mathcal{D}_{F_k} \quad : \quad \gamma_k \odot p(S) \odot \gamma_k^{(-1)}(a) \;\subseteq F(a)$,
ahol $\gamma_k = (\gamma_{k\,l_1} \ldots \gamma_{k\,l_{|L|}})$,
$\gamma_k \subseteq \left(\underset{l \in L_k}{\times} B_{il}\right) \times \left(\underset{l \in L_k}{\times} A_{k\,l},\right) \;\wedge\; \forall b_i \in \left(\underset{l \in L_k}{\times} B_{il}\right) : \gamma_k(b_i) = \underset{l \in L_k}{\times} \gamma_{k\,l}(b_i)$

$$\gamma_{k\,l}(b_i) \stackrel{\mathrm{def}}{=} \begin{cases} \varrho|_{[I]}(b_{il}) & \text{,ha } A_{k\,l} = T_S \\ \{b_{il}\} & \text{,ha } A_{k\,l} = B_{il} \end{cases}$$

SPECIFIKÁCIÓ

Definíciók

1.) Az R utófeltétel S programra vonatkozó ***leggyengébb előfeltétel***ének nevezzük azt az $lf(S,R)$ állítást, amelyre $\boxed{9}$
$[lf(S,R)] = \{a \in \mathcal{D}_{p(S)} \mid p(S)(a) \subseteq [R]\}$.

2.) $F^{-1} : 2^A \to 2^A, \qquad F^{-1}(H) = \{a \in A \mid F(a) = H\}$.

3.) B az $F \subseteq A \times A$ feladat ***paramétertere***, ha
$\exists\, F_1, F_2 : F_1 \subseteq A \times B\,,\; F_2 \subseteq B \times A$ és $F = F_2 \circ F_1$.

Tételek

1.)

a.) $[lf(S,R)] = p(S)^{-1}([R])$.

b.) $lf(S,R)(a) = a \in [R \circ p(S)]$.

2.) A leggyengébb előfeltétel tulajdonságai $\boxed{10}$

a.) $lf(S, hamis) = hamis$ (A csoda kizárásának elve).

b.) $lf(S, P \wedge Q) = lf(S,P) \;\wedge\; lf(S,Q)$.

c.) $lf(S, P \vee Q) \;\Leftarrow\; lf(S,P) \;\vee\; lf(S,Q)$.

d.) $Q \;\to\; R \;\Rightarrow\; lf(S,Q) \;\to\; lf(S,R)$ (Monotonitás).

3.) A tulajdonságok duálisai

a.) $\mathcal{D}_{p(S)} = \emptyset \Leftrightarrow lf(S,R) = hamis$.

b.) $S_1 \subseteq S_2 \Rightarrow [lf(S_2,R)] \subseteq [lf(S_1,R)]$.

c.) Ha $S_1 \cap S_2$ program, akkor $[lf(S_1,R)] \cap lf(S_2,R)] \subseteq [lf(S_1 \cap S_2, R)]$.

d.) $[lf(S_1 \cup S_2, R)] \subseteq [lf(S_1,R)] \cup [lf(S_2,R)]$.

4.) A leggyengébb előfeltétel és a megoldás kapcsolata $\boxed{11}$

a.) S megoldja F-t $\Leftrightarrow$ $\forall\, a \in \mathcal{D}_F \ :\ a \in [lf(S, P(F(a)))]$.

b.) S megoldja F-t $\Leftrightarrow$ $\forall\, H \subseteq \mathcal{D}_{F^{-1}} \ :\ F^{-1}(H) \subseteq [lf(S, P(H))]$.

5.) (*Specifikáció tétele*) $\boxed{12}$

Legyen $F \subseteq A \times A$, B az F egy paramétertere.
Legyen $b \in B$ és $[Q_b] = F_1{}^{(-1)}(b)$ és $[R_b] = F_2(b)$.
Ha $\forall\, b \in B$-re : $Q_b \Rightarrow lf(S, R_b)$, akkor S megoldja F-et.

6.) A specifikáció tétele nem megfordítható, azaz $\boxed{13}$

$\exists F \subseteq A \times A$ feladat, S program, $\exists F_1,\ F_2$, hogy $F = F_2 \circ F_1$ és Q_b, R_b adottak a specifikáció tétele szerint, és S megoldja F-t, és $\neg\forall b \in B : Q_b \Rightarrow lf(S, R_b)$.

ELEMI PROGRAMOK

Definíciók

1.) *Elem*inek nevezünk egy A állapottéren egy S *program*ot, ha
$\forall a \in A \ :\ S(a) \subseteq \{(a), (aaa\ldots), (a, b) \mid b \in A \wedge b \neq a\}$.

2.) *Üres*nek nevezzük, és SKIP-pel jelöljük azt a programot, amire
$\forall a \in A :\ SKIP(a) = \{(a)\}$.

3.) *Törlődés*nek nevezzük, és ABORT-tal jelöljük azt a programot, amire
$\forall a \in A :\ ABORT(a) = \{(aaa\ldots)\}$.

4.) Legyen $A = A_1 \times \ldots \times A_n$, $F = (F_1, \ldots, F_n)$, ahol $F_i \subseteq A \times A_i$. Az S program *általános értékadás*, ha $\boxed{14}$

$S = \{(a, red(a, b)) \mid a, b \in A \wedge a \in \bigcap_{i\in[1,n]} \mathcal{D}_{F_i} \wedge b \in F(a)\} \cup$

$\{(a, (aaa\ldots)) \mid a \in A \wedge a \notin \bigcap_{i\in[1,n]} \mathcal{D}_{F_i}\}$

Megjegyzés:

$\mathcal{D}_F = \bigcap_{i\in[1,n]} \mathcal{D}_{F_i}$

$F(a) = F_1(a) \times F_2(a) \times \ldots \times F_n(a)$

a) Ha $\mathcal{D}_F = A$, akkor az S programot *értékkiválasztás*nak nevezzük és $a :\in F(a)$ -val jelöljük.

b) Ha az F reláció függvény akkor az S programot *értékadás*nak nevezzük és $a := F(a)$-val jelöljük.

c) Ha $\mathcal{D}_F \subset A$, akkor *parciális értékkiválasztás*ról beszélünk.

d) Ha $\mathcal{D}_F \subset A$ és F determinisztikus, akkor S *parciális értékadás* .

e) Ha $\forall i, j \in [1..n] : i \neq j \wedge F_i(a_1, .., a_i, .., a_n) \neq a_i \Rightarrow F_j(a_1, .., a_j, .., a_n) = a_j$, akkor S *egyszerű* (általános) értékadás (egyébként *szimultán értékadás*).

Tételek

1.) $lf(SKIP, R) = R$.

2.) $lf(ABORT, R) = hamis$.

3.) Ha S az F reláció által meghatározott értékadás, akkor $lf(S, R) = F^{-1}([R])$.

PROGRAMKONSTRUKCIÓK

Definíciók

1.) Legyenek S_1,$S_2 \subseteq A \times A^{**}$ programok. Az $S \subseteq A \times A^{**}$ relációt az S_1 és S_2 ***szekvenciá***jának nevezzük, és $(S_1; S_2)$-vel jelöljük, ha $\boxed{15}$

$$\forall a \in A : S(a) = \{\alpha \mid \alpha \in S_1(a) \cap A^{\infty}\} \cup \\ \cup \{\chi_2(\alpha, \beta) \mid \alpha \in S_1(a) \cap A^* \wedge \beta \in S_2(\tau(\alpha))\}.$$

2.) Legyenek $\pi_1, \ldots, \pi_n$ feltételek, $S_1, \ldots, S_n$ programok A-n. Az $IF \subseteq A \times A^{**}$ relációt az S_i-kből képzett ***elágazás***nak nevezzük és $(\pi_1 : S_1, \ldots, \pi_n : S_n)$-nel jelöljük, ha $\boxed{16}$

$$\forall a \in A : IF(a) = (\bigcup_{i \in [1,n]} w_i(a)) \cup w(a),$$

ahol

$$w_i(a) = \begin{cases} S_i(a) \ , & \text{ha } a \in [\pi_i] \\ \emptyset \ , & \text{különben} \end{cases}$$

$$w(a) = \begin{cases} \{(a, a, \ldots)\} \ , & \text{ha } a \notin \bigcup_{i \in [1,n]} [\pi_i] \\ \emptyset \ , & \text{különben} \end{cases}$$

3.) Legyen π feltétel és S_0 program A-n. A $DO \subseteq A \times A^{**}$ relációt az S_0-ból és π-ből képezett ***ciklus***nak nevezzük, és (π, S_0)-lal jelöljük, ha $\boxed{17}$

$\forall a \in [\neg\pi] : DO(a) = \{(a)\}$,

$\forall a \in [\pi] : DO(a) = \{\alpha \mid \alpha \in A^{**} \wedge \exists \alpha_1, \ldots, \alpha_n \in A^{**} :$
$(\alpha = \chi_n(\alpha_1, \ldots, \alpha_n) \wedge \alpha_1 \in S_0(a) \wedge \forall i \in [1, n-1] : \alpha_i \in A^* \wedge \alpha_{i+1} \in S_0(\tau(\alpha_i)) \wedge \tau(\alpha_i) \in [\pi]) \wedge ((\alpha_n \in A^* \wedge \tau(\alpha_n) \notin [\pi]) \vee (\alpha_n \in A^{\infty}))\} \ \cup$
$\{\alpha \mid \alpha \in A^{\infty} \wedge \exists \alpha_1, \ldots, \alpha_k, .. \in A^* : (\alpha = \chi_{\infty}(\alpha_1, \ldots, \alpha_k, ..) \wedge \alpha_1 \in S_0(a) \wedge$
$\forall i \in \mathcal{N} : \alpha_i \in A^* \wedge \alpha_{i+1} \in S_0(\tau(\alpha_i)) \wedge \tau(\alpha_i) \in [\pi])\}$.

Tételek

1.) A szekvencia, az elágazás és a ciklus program. $\boxed{18}$

2.) Legyen $S = (S_1; S_2)$ szekvencia, $IF = (\pi_1 : S_1, \ldots, \pi_n : S_n)$ elágazás és $DO = (\pi, S_0)$ ciklus. Ekkor $\boxed{19}$

a.) $p((S_1; S_2)) = p(S_2) \odot p(S_1)$.

b.) $\mathcal{D}_{p(IF)} = \{a \in A \mid a \in \bigcup_{i \in [1,n]} [\pi_i] \wedge \forall i \in [1, n] : a \in [\pi_i] \Rightarrow a \in \mathcal{D}_{p(S_i)}\}$

$\forall a \in \mathcal{D}_{p(IF)} : p(IF)(a) = \bigcup_{i \in [1,n]} pw_i(a)$, ahol

$$pw_i(a) = \begin{cases} p(S_i)(a), & \text{ha } a \in [\pi_i] \\ \emptyset, & \text{különben} \end{cases}$$

c.) $p(DO(\pi, S_0)) = \overline{p(S_0)|_{\pi}}$.

3.) A szekvencia, az elágazás és a ciklus leggyengébb előfeltétele rendelkezik a következő tulajdonságokkal :

a) Legyen $S = (S_1; S_2)$, és Q, R olyan állítások A-n, amelyekre $Q \Rightarrow lf(S, R)$. Ekkor létezik olyan Q' állítás, amire $Q \Rightarrow lf(S_1, Q') \;\wedge\; Q' \Rightarrow lf(S_2, R)$.

b) Legyen $IF = (\pi_1 : S_1, \ldots, \pi_n : S_n)$, és Q, R olyan állítások A-n, amelyekre $Q \wedge \left(\bigvee_{i\in[1,n]} \pi_i\right) \Rightarrow lf(IF, R)$. Ekkor $\forall i \in [1, n] : Q \wedge \pi_i \Rightarrow lf(S_i, R)$.

c) Legyen $DO = (\pi, S_0)$, és Q, R olyan állítások A-n, amelyekre $Q \Rightarrow lf(DO, R)$, és tegyük fel, hogy $\forall a \in \mathcal{D}_{p(S_0)} \cap [\pi] \;:\; \exists k \in \mathcal{N} \;:\; p(S_0)^k(a) = \emptyset$. Ekkor létezik P állítás és $t : A \to Z$ függvény, amelyekre

1. $Q \Rightarrow P$,
2. $P \wedge \neg\pi \Rightarrow R$,
3. $P \wedge \pi \Rightarrow t > 0$,
4. $P \wedge \pi \wedge t = t' \Rightarrow lf(S_0, P \wedge (t \leq t' - 1))$.

LEVEZETÉSI SZABÁLYOK

Tételek

1.) A szekvencia levezetési szabálya. $\boxed{20}$
Legyen $S = (S_1; S_2)$, és adott Q, R és Q' állítás A-n. Ha

1. $Q \Rightarrow lf(S_1, Q')$ és
2. $Q' \Rightarrow lf(S_2, R)$, akkor $Q \Rightarrow lf(S, R)$.

2.) Az elágazás levezetési szabálya. $\boxed{21}$
Legyen $IF = (\pi_1 : S_1, \ldots, \pi_n : S_n)$, és adott Q, R állítás A-n. Ha $\forall i \in [1, n] : Q \wedge \pi_i \Rightarrow lf(S, R)$, akkor $\mathrm{Q} \wedge \left(\bigvee_{\mathrm{i}\in[1,\mathrm{n}]} \pi_{\mathrm{i}}\right) \Rightarrow \mathrm{lf(IF, R)}$.

3.) A ciklus levezetési szabálya. $\boxed{22}$
Legyen adott P, Q, R állítás A-n, $t : A \to Z$ függvény és legyen $DO = (\pi, S_0)$. Ha

1. $Q \Rightarrow P$,
2. $P \wedge \neg\pi \Rightarrow R$,
3. $P \wedge \pi \Rightarrow t > 0$,
4. $P \wedge \pi \wedge t = t' \Rightarrow lf(S_0, P \wedge (t \leq t' - 1))$ akkor $Q \Rightarrow lf(DO, R)$.

A t függvényt *variáns függvény*nek, a P állítást *invariáns* tulajdonságnak nevezzük.

TÍPUSKONSTRUKCIÓK

Legyenek a $T_0, T_1, \ldots, T_n, E_1, \ldots, E_n$ nem üres, véges vagy megszámlálható halmazok, $E = \bigcup_{i=1,n} E_i$, $\quad \varrho_i \subset E^* \times T_i$ reprezentációs függvények, $\mathcal{T}_i = (\varrho_i, I_i, \mathcal{S}_i)$ $(i = 0, \ldots, n)$ típusok.

Definíciók

1.) Direktszorzat

$\varrho = \psi_D \circ \varphi_D$, ahol $\psi_D \subset B \times T, B = T_1 \times \ldots \times T_n$, $\varphi_D = \{(\varepsilon, b) \in E^* \times B \mid \exists \varepsilon_1, \ldots, \varepsilon_n \in E^* : \forall i \in [1,n] : (\varepsilon_i, b_i) \in \varrho_i \wedge \varepsilon = kon(\varepsilon_1, \ldots, \varepsilon_n)\}$.

Jelölés: $\mathcal{T} = (\mathcal{T}_1, \ldots, \mathcal{T}_n)$.

Ha ψ_D kölcsönösen egyértelmű, akkor $\mathcal{T}$ rekord típus. $\boxed{23}$

2.) Unió

$\varrho = \psi_U \circ \varphi_U$, ahol $\psi_U \subset B \times T, B = T_1 \cup \ldots \cup T_n$,

$\varphi_U = \{(\varepsilon, b) \in E^* \times B \mid \exists i \in [1,n] : (\varepsilon, b) \in \varrho_i\}$.

Jelölés: $\mathcal{T} = (\mathcal{T}_1; \ldots; \mathcal{T}_n)$.

Ha ψ_U kölcsönösen egyértelmű, akkor $\mathcal{T}$ egyesítés.

3.) Iterált

$\varrho = \psi_I \circ \varphi_I$, ahol $\psi_I \subset T_0^* \times T$, $\varphi_I = \{(\varepsilon, \beta) \in E^* \times T_0^* \mid \exists \alpha \in (E^*)^* : \mid \alpha \mid = \mid \beta \mid \wedge \forall i \in [1, \mid \alpha \mid] : (\alpha_i, \beta_i) \in \varrho_0 \wedge \varepsilon = kon(\alpha_1, \ldots, \alpha_{|\alpha|}\}$.

Jelölés: $\mathcal{T} = it(\mathcal{T}_0)$.

Ha ψ_I kölcsönösen egyértelmű, akkor $\mathcal{T}$ sorozat típus.

Jelölés: $\mathcal{T} = seq(\mathcal{T}_0)$.

Ha $D_{\psi_I} = T_0^*$ és $(\alpha, t), (\beta, t) \in \psi_I \equiv \alpha \in perm(\beta)$, akkor $\mathcal{T}$ kombináció típus.

Jelölés: $\mathcal{T} = com(\mathcal{T}_0)$.

Ha $D_{\psi_I} = T_0^*$ és $(\alpha, t), (\beta, t) \in \psi_I \equiv \cup\{\alpha_i\} = \cup\{\beta_i\}$, akkor $\mathcal{T}$ halmaz típus.

Jelölés: $\mathcal{T} = set(\mathcal{T}_0)$.

1. Specifikációs függvények

Definíciók

1. Szelekciós függvények
   a) Legyen $T = (T_1, ..., T_n)$ rekord típus. $s_1, ..., s_n$ a rekord szelektorfüggvényei, ha $s_i : T \to T_i$, és $\forall t \in T$ -re $\lambda_D \circ (s_1, ..., s_n)(t) = t$.

      A szelektorokkal ellátott rekord jelölése: $T = (s_1 : T_1, ..., s_n : T_n)$.

   b) Legyen $T = (T_1; ...; T_n)$ egyesítés típus. $s_1, ..., s_n$ az egyesítés szelektorfüggvényei, ha $s_i : T \to L$ és $\forall t \in T$ -re $s_i(t) \equiv t \in \lambda_U(T_i)$.

      A szelektorokkal ellátott egyesítés jelölése: $T = (s_1 : T_1; ...; s_n : T_n)$.

   c) Legyen $T = it(T_0)$ iterált típus, $t_0 \in T_0, t \in T$. A $t_0 \in t$ relációt úgy értelmezzük, hogy $t_0 \in t \equiv \exists (\alpha, t) \in \lambda_I, i \in N : \alpha_i = t_0$.

   d) $T = seq(T_0)$. $kiv : T \times N \to T_0$ a sorozatra értelmezett szelekciós függvény, ha

$$kiv(t,i) \stackrel{\text{def}}{=} \begin{cases} \alpha & \text{,ha } 1 \leq i \leq dom(t) \ \wedge \ (\alpha, t) \in \lambda \\ \text{nem definiált} & \text{, egyébként.} \end{cases}$$

      A sorozatra vonatkozó kiválasztásfüggvényt indexeléssel jelöljük: t_j.

2. A $T = it(T_0)$ iterált típushoz definiáljuk a $dom : T \to N_0$ függvényt.

Legyen $(\alpha, t) \in \lambda_I$, ekkor

$$dom(t) \overset{\text{def}}{=} \begin{cases} \mid \alpha \mid & \text{ha } T \text{ sorozat, vagy kombináció} \\ \mid \cup_{i=1,|\alpha|}\{\alpha_i\} \mid & \text{ha } T \text{ halmaz} \\ \text{definiálatlan} & \text{egyébként} \end{cases}$$

3. Sorozatra értelmezett függvények
   Legyen $T = seq(T_0) \ \wedge \ (\alpha, t) \in \lambda_I$.
   a) $low : T \to T_0$. $low(t) \overset{\text{def}}{=} \alpha_1$, ha $dom(t) \neq 0$.
   b) $hiw : T \to T_0$. $hiw(t) \overset{\text{def}}{=} \alpha_{|\alpha|}$, ha $dom(t) \neq 0$.
   b) $ext : T \times T \to T$. Ha $(\alpha_1, t_1), (\alpha_2, t_2) \in \lambda_I$, akkor
   $ext(t_1, t_2) \overset{\text{def}}{=} \lambda_I(kon(\alpha_1, \alpha_2))$.
   $ext(t_1, ..., t_k) \overset{\text{def}}{=} ext(ext(t_1, ..., t_{k-1}), t_k)$.
   c) $rem : T \times T \to T$.

$$rem(t_1, t_2) \overset{\text{def}}{=} \begin{cases} ext(t', t'') & \text{, ha } \exists t', t'' : t_1 = ext(t', t_2, t'') \\ t_1 & \text{, egyébként} \end{cases}$$

   d) $loext : T \times T_0 \to T$. $loext(t, t_0) \overset{\text{def}}{=} ext(\lambda_I(t_0), t)$.
   e) $hiext : T \times T_0 \to T$. $hiext(t, t_0) \overset{\text{def}}{=} ext(t, \lambda_I(t_0))$.
   f) $hirem : T \to T$. $hiext(hirem(t), hiw(t)) = t$, ha $dom(t) \neq 0$.
   g) $lorem : T \to T$. $loext(lorem(t), low(t)) = t$, ha $dom(t) \neq 0$.

4. A kombinációra értelmezett függvények
   Legyen $T = com(T_0)$.
   a) $ism : T \times T_0 \to N_0$. Ha $\alpha \in \lambda_I^{-1}(t)$ és $t_0 \in T_0$, akkor

$$ism(t, t_0) \overset{\text{def}}{=} \sum_{i=1}^{|\alpha|} \chi(\alpha_i = t_0)$$

   ahol $\chi : L \to \{0, 1\}$ ún. karakterisztikus függvény: $\chi(\uparrow) = 1, \chi(\downarrow) = 0$.

5. A függvény típus
   Legyen H tetszőleges rendezett halmaz, amelyen értelmezve vannak a $pred, succ$ és ord függvények, $ord(succ(x)) - ord(x) = 1$. Legyen $T = seq(T_0)$. Az $F = (H, T)$ rekord típust függvény típusnak nevezzük. Jele: $\mathcal{F} = fun(H, T_0)$.

6. Függvény típuson értelmezett függvények
   Legyen $\mathcal{F} = fun(H, T_0)$ függvény típus, továbbá $((m, t), f) \in \lambda_D$ tetszőleges rögzített elem. Ekkor
   a) $dom : F \to N_0, dom(f) \overset{\text{def}}{=} dom(t)$.

b) $lob : F \to H, lob(f) \stackrel{\text{def}}{=} m$.

c) $hib : F \to H, ord(hib(f)) - ord(lob(f)) = dom(f) - 1$.

d) $kiv : F \times H \to T_0$.

e) $kiv(f,i) \stackrel{\text{def}}{=} t_{ord(i)-ord(lob(f))+1}$, ha $lob(f) \leq i \leq hib(f)$.

f) $low : F \to T_0, low(f) \stackrel{\text{def}}{=} t_1$, ha $dom(f) \neq 0$.

g) $hiw : F \to T_0, hiw(f) \stackrel{\text{def}}{=} t_{dom(t)}$, ha $dom(f) \neq 0$.

h) $hiext : F \times T_0 \to F \ hiext(f, t_0) \stackrel{\text{def}}{=} \lambda_D(m, ext(t, t_0))$.

i) $loext : F \times T_0 \to F \ loext(f, t_0) \stackrel{\text{def}}{=} \lambda_D(m-1, ext(t_0, t))$.

k) $hirem : F \to F$, $hirem(f) \stackrel{\text{def}}{=} \lambda_D(m, hirem(t))$, ha $dom(f) \neq 0$.

l) $hirem : F \to F$, $lorem(f) \stackrel{\text{def}}{=} \lambda_D(m+1, lorem(t))$, ha $dom(f) \neq 0$.

2. Nevezetes típusok és típusosztályok

A továbbiakban R jelöljön tetszőleges utófeltételt.

1. Rekord típusok műveletei: $t_i := t.s_i$, $t.s_i := t_i$.

   Legyen $A = T \times T_1 \times ... \times T_n$ és $t : T, t_i : T_i (i \in [1..n])$ és $T = (s_1 : T_1, ..., s_n : T_n)$.

   a) $t_i := t.s_i$, ahol $i \in [1, ..., n])$. $lf(t_i := t.s_i, R) = R^{t_i \leftarrow t_i.s}$.

   b) $t.s_i := t_i$, ahol $i \in [1, ..., n])$. $lf(t.s_i := t_i, R) = R^{t \leftarrow t'}$,
   ahol $t' \in T$ és $s_j(t') = t.s_j$, ha $i \neq j, s_i(t') = t_i$.

2. Egyesítés típusok műveletei: $t.s_i$ logikai függvények.

   Legyen $T = (s_1 : T_1; ...; s_n : T_n)$.

   a) $i \in [1, ..., n])$. $t.s_i \equiv t \in \lambda_U(T_i)$.

3. Halmaz típusok műveletei: $t_0 \in t, t_0 : mem(t), t := t \tilde{-} \{t_0\}, t := t \tilde{\cup} \{t_0\}$.

   Legyen $T = set(T_0)$.

   a) $t_0 : mem(t)$. $lf(t_0 : mem(t), R) = dom(t) \neq \emptyset \wedge forall e \in t : R^{t_0 \leftarrow e}$.

   b) $t := t \tilde{-} \{t_0\}$. $lf(t := t \tilde{-} t_0, R) = t_0 \in t \wedge R^{t \leftarrow t - \{t_0\}}$.

   c) $t := t \tilde{\cup} \{t_0\}$ $lf(t := t \tilde{\cup} t_0, R) = t_0 \in t \wedge R^{t \leftarrow t \cup \{t_0\}}$.

   d) $t_0 \in t$ logikai függvény.

4. Kombináció típusok műveletei: $t_0 \in t, t_0 : mem(t), t := t \tilde{-} t_0, t := t \tilde{\cup} t_0, m : ism(t, t_0)$.

   Legyen $T = com(T_0)$.

   a)-d) $t_0 \in t, t_0 : mem(t), t := t \tilde{-} t_0, t := t \tilde{\cup} t_0, m : ism(t, t_0)$.

   (A műveletek specifikációja megegyezik a halmaz típus megfelelő műveleteinek specifikációjával.)

e) $lf(m : ism(t,t_0),R) = R^{m\leftarrow ism(t_0,t)}$.

5. Sorozat típusok műveletei: $dom, low, hiw, hiext, loext, lorem, hirem, t_0 := t_i, t_i := t_0, lopop, hipop$.

   Legyen $T = seq(T_0)$, $A = T \times T_0$, $t_0 : T_0, t : T, m : N_0$.

   a) $lf(m := t.dom, R) = R^{m\leftarrow dom(t)}$.
   b) $lf(t_0 := t.low, R) = dom(t) \neq 0 \wedge R^{t_0\leftarrow low(t)}$.
   c) $lf(t_0 := t.hiw, R) = dom(t) \neq 0 \wedge R^{t_0\leftarrow hiw(t)}$.
   d) $lf(t : hiext(t_0), R) = R^{t\leftarrow hiext(t,t_0)}$.
   e) $lf(t : loext(t_0), R) = R^{t\leftarrow loext(t,t_0)}$.

   Legyen $A = T$ és $t : T$.

   f) $lf(t : lorem, R) = dom(t) \neq 0 \wedge R^{t\leftarrow lorem(t)}$.
   g) $lf(t : hirem, R) = dom(t) \neq 0 \wedge R^{t\leftarrow hirem(t)}$.

   Legyen $A = T \times T_0 \times N$ és $t_0 : T_0, t : T, i : N$.

   h) $lf(t_0 := t_i, R) = 1 \leq i \leq dom(t) \wedge R^{t_0\leftarrow t_i}$.
   i) $lf(t_i := t_0, R) = 1 \leq i \leq dom(t) \wedge R^{t\leftarrow t'}$,
   ahol $t' \in T, t'_j = t_j$ ha $i \neq j$ és $t'_i = t_0$.

   Legyen $A = T \times T_0$ és $t_0 : T_0, t : T$.

   j) $lf(t_0, t : lopop, R) = dom(t) \neq 0 \wedge R^{t_0,t\leftarrow low(t),lorem(t)}$,
   $t_0, t : lopop \equiv t_0, t := t.low, t : lorem$.
   k) $lf(t_0, t : hipop, R) = dom(t) \neq 0 \wedge R^{t_0,t\leftarrow hiw(t),hirem(t)}$,
   $t_0, t : hipop \equiv t_0, t := t.low, t : hirem$.

6) A függvény típusok műveletei: $dom, lob, hib, hiw, low, t_0 := f_k, f_k := t_0, hiext, loext, hirem, lorem, hipop, lopop$.
   Legyen $\mathcal{F} = fun(H, T_0)$.

   a) $lf(k := f.dom, R) = R^{k\leftarrow dom(f)}$.
   b) $lf(k := f.lob, R) = R^{k\leftarrow lob(f)}$.
   c) $lf(k := f.hib, R) = R^{k\leftarrow hib(f)}$.
   d) $lf(t_0 := f.low, R) = dom(f) \neq 0 \wedge R^{t_0\leftarrow low(f)}$.
   e) $lf(t_0 := f.hiw, R) = dom(f) \neq 0 \wedge R^{t_0\leftarrow hiw(f)}$.
   f) $lf(t_0 := f_k, R) = lob(f) \leq k \leq hib(f) \wedge R^{t_0\leftarrow f_k}$.
   g) $lf(f_k := t_0, R) = lob(f) \leq k \leq hib(f) \wedge R^{f\leftarrow f'}$,
   ahol $f' \in F$ és $f'_i = f_i$, ha $i \neq k$ és $f'_k = t_0$.
   h) $lf(f : hiext(t_0), R) = R^{f\leftarrow hiext(f,t_0)}$.
   i) $lf(f : loext(t_0), R) = R^{f\leftarrow loext(f,t_0)}$.
   j) $lf(f : hirem, R) = dom(f) \neq 0 \wedge R^{f\leftarrow hirem(f)}$.
   k) $lf(f : lorem, R) = dom(f) \neq 0 \wedge R^{f\leftarrow lorem(f)}$.

l) $lf(t_0, f : hipop, R) = dom(f) \neq 0 \wedge R^{t_0, f \leftarrow hiw(f), hirem(f)}$.

m) $lf(t_0, f : lopop, R) = dom(f) \neq 0 \wedge R^{t_0, f \leftarrow low(f), lorem(f)}$.

7. Szekvenciális input file
   A $SeqInp(T_0) = (seq(T_0); reset, st, dt, t : read)$ típust szekvenciális input file-nak nevezzük, ahol
   $lf(st, dt, t : read, R) = dom(t) \neq 0 \Rightarrow R^{st, dt, t \leftarrow norm, low(t), lorem(t)} \wedge$
   $dom(t) = 0 \Rightarrow R^{st \leftarrow abnorm}$.

8. Szekvenciális output file
   A $SeqOut(T_0) = (seq(T_0); create, t : write(t_0))$ típust szekvenciális output file-nak nevezzük, ahol
   $t : write(t_0) \stackrel{\text{def}}{=} t : hiext(t_0)$.

9. A verem típus
   A $Verem(T_0) = (seq(T_0); create, pop, push, top, empty)$ típust verem típusnak nevezzük.
   Legyen $v : Verem, t_0 : T_0$.
   a) $pop(v) \stackrel{\text{def}}{=} v : hirem$.
   b) $push(v, t_0) \stackrel{\text{def}}{=} v : hiext(t_0)$.
   c) $top(v, t_0) \stackrel{\text{def}}{=} t_0 := v.hiw, \quad empty(v) \equiv v.dom = 0$.

10. A sor típus
   A $Sor(T_0) = (seq(T_0); create, pop, push, top, empty)$ típust sor típusnak nevezzük.
   Legyen $s : Sor, t_0 : T_0$.
   a) $pop(s) \stackrel{\text{def}}{=} s : lorem$.
   b) $push(s, t_0) \stackrel{\text{def}}{=} s : hiext(t_0)$.
   c) $top(s, t_0) \stackrel{\text{def}}{=} t_0 := s.low$.
   d) $empty(s) \equiv s.dom = 0$.

11. A vektor típus
   A $T = (fun(H, T_0); dom, lob, hib, low, hiw, t_0 := t_k, t_k := t_0)$ típust vektor típusnak nevezzük.
   A vektor típus jele: $Vektor(H, T_0)$.

PROGRAMOZÁSI TÉTELEK

1. Lineáris keresés

A feladat az, hogy az egész számok egy zárt intervallumán keressük meg a legkisebb adott tulajdonságú egész számot. $\boxed{24}$

$$A = \mathcal{Z} \times \mathcal{Z} \times \mathcal{Z}, \qquad m, n, i : \mathcal{Z}.$$
$$B = \mathcal{Z} \times \mathcal{Z}, \qquad m', n' : \mathcal{Z}.$$
$$Q_{m',n'} = (m = m' \wedge n = n' \wedge [m,n] \cap [\beta] \neq \emptyset),$$
$$R_{m',n'} = (Q_{m',n'} \wedge i \in [m,n] \wedge \beta(i) \wedge \forall j \in [m,i) : \neg\beta(j)).$$

i.) $P_{m',n'} = (Q_{m',n'} \wedge i \in [m,n] \wedge \forall j \in [m,i) : \neg\beta(j))$.

$i := m$
$\neg\beta(i)$
$i := i + 1$

ii.) $P'_{m',n'} = (Q_{m',n'} \wedge i \in [m-1,n] \wedge (\forall j \in [m,i) : \neg\beta(j)) \wedge \neg l = \forall j \in [m,i] : \neg\beta(j))$.

$i, l := m - 1, \downarrow$
$\neg l$
$l := \beta(i + 1)$
$i := i + 1$

iii.) Tegyük fel, hogy $\beta = \gamma \vee \delta$ és $[\delta] \cap [m,n] \neq \emptyset$. Keressük meg a legkisebb $i \in [m,n] \cap [\beta]$ elemet, és mondjuk meg azt is, hogy $\gamma(i)$ igaz-e !

$$A = \mathcal{Z} \times \mathcal{Z} \times \mathcal{Z} \times \mathcal{L}, \qquad m, n, i : \mathcal{Z}, \qquad u : \mathcal{L}.$$
$$B = \mathcal{Z} \times \mathcal{Z}, \qquad m', n' : \mathcal{Z}.$$
$$Q_{m',n'} = (m = m' \wedge n = n' \wedge [m,n] \cap [\delta] \neq \emptyset),$$
$$R_{m',n'} = (Q_{m',n'} \wedge u = (\exists j \in [m,n] : \gamma(j) \wedge \forall k \in [m, j-1] : \neg\delta(k)) \wedge$$
$$u \Rightarrow (i \in [m,n] \wedge \gamma(i) \wedge \forall j \in [m, i-1] : \neg\gamma(j)).$$
$$P_{m',n'} = (Q_{m',n'} \wedge i \leq n \wedge \neg u = \forall j \in [m,i] : \neg\gamma(j) \wedge$$
$$\neg v = \forall j \in [m,i] : \neg\delta(j) \wedge \forall j \in [m,i) : \neg\beta(j)).$$

$i, u, v := m - 1, \downarrow, \downarrow$
$\neg u \wedge \neg v$
$u, v := \gamma(i + 1), \delta(i + 1)$
$i := i + 1$

2. Logaritmikus keresés

Legyen H egy olyan halmaz, amin értelmezve van egy rendezési reláció. Legyen $f : \mathcal{Z} \to H$ monoton növekedő függvény. A feladat az, hogy eldöntsük egy adott $h \in H$

értékről, hogy felveszi-e azt az f függvény egy adott $[m,n] \subset \mathcal{Z}$ intervallumon, és ha igen, akkor adjuk meg egy olyan pontját, ahol a függvényérték h. $\boxed{25}$

$$\begin{aligned}
&A = \mathcal{Z} \times \mathcal{Z} \times \mathcal{Z} \times H \times \mathcal{L}, \qquad m,n,i : \mathcal{Z}, \quad h : H, \quad l : \mathcal{L}.\\
&B = \mathcal{Z} \times \mathcal{Z} \times H, \qquad m',n' : \mathcal{Z}, \quad h' : H.\\
&Q_{m',n',h'} = (m = m' \land n = n' \land h = h' \land \forall k,j \in [m,n] : k \leq j \Rightarrow f(k) \leq f(j)),\\
&R_{m',n',h'} = (Q_{m',n',h'} \land l = (\exists j \in [m,n] : f(j) = h)\\
&\qquad\qquad \land\, l \Rightarrow (i \in [m,n] \land f(i) = h)).\\
&P_{m',n',h'} = (Q_{m',n',h'} \land [u,v] \subseteq [m,n]\\
&\qquad\qquad \land\, \forall j \in [m,n] \setminus [u,v] : f(j) \neq h \land l = (i \in [u,v] \land f(i) = h)).
\end{aligned}$$

$u,v,i,l := m,n,m-1,\downarrow$		
$\neg l \land (u \leq v)$		
$i := [(u+v)/2]$		
$f(i) < h$	$f(i) > h$	$f(i) = h$
$u := i+1$	$v := i-1$	$l :=\uparrow$

3. Maximumkeresés

Legyen H egy tetszőleges rendezett halmaz és $f : \mathcal{Z} \to H$ egy függvény. Feladatunk az, hogy egy adott $[u,v] \subset \mathcal{Z}$ intervallumban keressük meg az f függvény maximumát és egy olyan helyét, ahol ezt a maximumértéket felveszi. $\boxed{26}$

$$\begin{aligned}
&A = H \times \mathcal{Z} \times \mathcal{Z} \times \mathcal{Z}, \qquad m : H, \qquad u,v,i : \mathcal{Z}.\\
&B = \mathcal{Z} \times \mathcal{Z}, \qquad u',v' : \mathcal{Z}.\\
&Q_{u',v'} = (u = u' \land v = v' \land u \leq v),\\
&R_{u',v'} = (Q_{u',v'} \land i \in [u,v] \land m = f(i) \land \forall j \in [u,v] : f(j) \leq f(i)).\\
&P_{u',v'} = (Q_{u',v'} \land k \in [u,v] \land i \in [u,k] \land m = f(i) \land \forall j \in [u,k] : f(j) \leq f(i)).
\end{aligned}$$

$i,k,m := u,u,f(u)$	
$k \neq v$	
$m < f(k+1)$	
$i,m := k+1, f(k+1)$	$SKIP$
$k := k+1$	

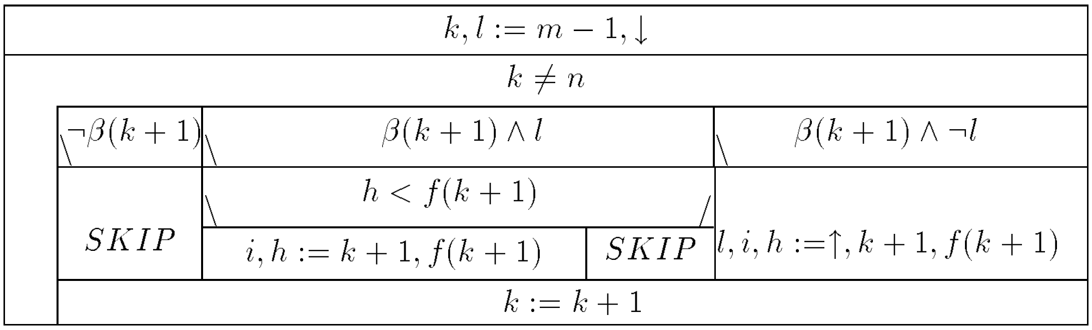
4. Feltételes maximumkeresés

Legyen H egy tetszőleges rendezett halmaz és $f : [m,n] \subset \mathcal{Z} \to H$ egy függvény. Legyen β egy az egész számokon értelmezett logikai függvény. Határozzuk meg a $[\beta] \cap [m,n]$ halmaz felett az f függvény maximumát és a halmaz azon elemét, amelyen

f a maximumértékét felveszi.

$$
\begin{aligned}
&A = \mathcal{Z} \times \mathcal{Z} \times \mathcal{Z} \times H \times \mathcal{L}, \qquad m, n, i : \mathcal{Z}, \qquad h : H \qquad l : \mathcal{L}.\\
&B = \mathcal{Z} \times \mathcal{Z}, \qquad m', n' : \mathcal{Z}.\\
&Q_{m',n'} = (m = m' \land n = n' \land m \leq n + 1),\\
&R_{m',n'} = (Q_{m',n'} \land l = (\exists i \in [m, n] : \varepsilon(m, n, i)) \land l \Rightarrow \varepsilon(m, n, i) \land h = f(i))).\\
&P_{m',n'} = (Q_{m',n'} \land u \in [m, n] \land\\
&\qquad\qquad l = (\exists i \in [m, k] : \varepsilon(m, k, i)) \land l \Rightarrow \varepsilon(m, k, i) \land h = f(i))),
\end{aligned}
$$

ahol $\quad \varepsilon(m, k, i) = i \in [m, k] \land \beta(i) \land \forall j \in [m, k] : \beta(j) \Rightarrow f(i) \geq f(j).$



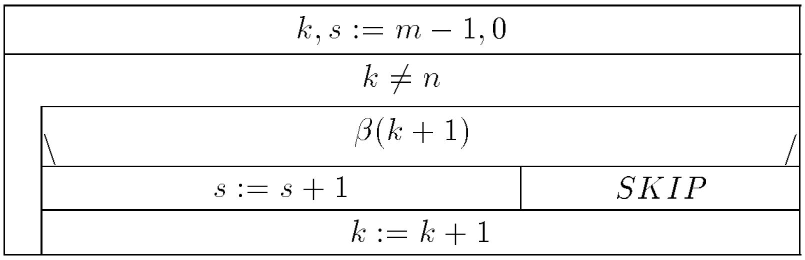
5. Adott tulajdonsággal rendelkező elemek megszámolása

Legyen β egy az egész számokon értelmezett logikai függvény. A feladat az, hogy számoljuk meg, hány helyen igaz β az $[m, n] \subset \mathcal{Z}$ intervallumban. $\boxed{27}$

$$
\begin{aligned}
&A = \mathcal{Z} \times \mathcal{Z} \times \mathcal{N}_0, \qquad m, n : \mathcal{Z}, \qquad s : \mathcal{N}_0.\\
&B = \mathcal{Z} \times \mathcal{Z}, \qquad m', n' : \mathcal{Z}.\\
&Q_{m',n'} = (m = m' \land n = n' \land m \leq n + 1),\\
&R_{m',n'} = (Q_{m',n'} \land s = \sum_{i=m}^{n} \chi(\beta(i))),
\end{aligned}
$$

ahol $\chi : \mathcal{L} \to \mathcal{N}_0, \chi(\uparrow) = 1, \chi(\downarrow) = 0.$

$$P_{m',n'} = (Q_{m',n'} \land k \leq n \land s = \sum_{i=m}^{k} \chi(\beta(i))).$$



6. Összegzés

Legyen H egy tetszőleges halmaz, amelyen értelmezett az összeadás művelete. Tegyük fel, hogy létezik neutrális eleme a H-nak az összeadásra nézve, valamint adott az $f : \mathcal{Z} \to H$ függvény. Feladatunk az, hogy egy adott $[m, n] \subset \mathcal{Z}$ intervallumban összegezzük az f függvény értékeit.

$$A = \mathcal{Z} \times \mathcal{Z} \times H, \quad m, n : \mathcal{Z}, \quad s : H.$$
$$B = \mathcal{Z} \times \mathcal{Z}, \quad m', n' : \mathcal{Z}.$$
$$Q_{m',n'} = (m = m' \wedge n = n' \wedge m \leq n + 1),$$
$$R_{m',n'} = (Q_{m',n'} \wedge s = \sum_{i=m}^{n} f(i)).$$
$$P_{m',n'} = (Q_{m',n'} \wedge k \leq n \wedge s = \sum_{i=m}^{k} f(i)).$$

$k, s := m - 1, 0$	
$k \neq n$	
	$s := s + f(i + 1)$
	$k := k + 1$

7. Rendezés

Legyen T tetszőleges rendezett típus és $X = seq(T)$, legyen továbbá $x : X$. Rendezzük át az x sorozatot úgy, hogy az elemei növekedő sorrendben legyenek. [28]

$$A = X, \quad x : X.$$
$$B = X, \quad x' : X.$$
$$Q_{x'} = (x = x'),$$
$$R_{x'} = (x \in perm(x') \wedge \varrho(x, 0, x.dom)),$$

ahol $\varrho(x, i, j) = (\forall k, l(i < k < l \leq j) : x_k \leq x_l),$

$$\mu(x, i, j) = (\forall k, l(1 \leq k \leq i < l \leq j) : x_k \leq x_l).$$

$$P_{x'} = (x \in perm(x') \wedge 0 \leq k \leq x.dom \wedge \varrho(x, k, x.dom) \wedge \mu(x, k, x.dom)).$$

$k := x.dom$	
$k \neq 0$	
	$UJSZELET$
	$k := m$

$P' = (\varrho(x, m, l) \wedge \mu(x, m, l) \wedge m < l \wedge 1 \leq l \leq k \wedge P_{x'}).$

$UJSZELET$

$m, l := 0, 1$	
$l \neq k$	
$x_{l+1} < x_l$	
$m, x_l, x_{l+1} := l, x_{l+1}, x_l$	$SKIP$
$l := l + 1$	

8. Speciális formában adott függvény értékének kiszámítása adott helyen

8.1. Függvénykompozícióval definiált függvény

Legyen H_1, H_2, H_3 tetszőleges halmaz, továbbá $f : H_1 \to H_3, g : H_1 \to H_2, h : H_2 \to H_3$, olyan függvények, hogy $f = h \circ g$. Feladatunk az, hogy meghatározzuk f függvény $i \in H_1$ helyen felvett értékét.

$$A = H_1 \times H_3, \quad i : H_1, \quad z : H_3.$$
$$B = H_1, \quad i' : H_1.$$
$$Q_{i'} = (i = i'),$$
$$R_{i'} = (Q_{i'} \wedge z = f(i)).$$
$$Q'_{i'} = (Q_{i'} \wedge z = g(i)).$$

$z := g(i)$
$z := h(z)$

8.2. Kapcsoszárójellel definiált függvény

Legyen H egy tetszőleges halmaz, továbbá $f, h_1, ..., h_k : \mathcal{Z} \to H$, olyan függvények, hogy

$$f(i) = \begin{cases} h_1(i), & \text{ha } \pi_1(i) \\ ... & \\ h_k(i), & \text{ha } \pi_k(i) \end{cases}$$

Feladatunk az, hogy meghatározzuk f függvény i helyen felvett értékét.

$$A = \mathcal{Z} \times H, \quad i : \mathcal{Z}, \quad z : H.$$
$$B = \mathcal{Z}, \quad i' : \mathcal{Z}.$$
$$Q_{i'} = (i = i') \wedge i \in \cup_{j=1}^{k} [\pi_j]),$$
$$R_{i'} = (Q_{i'} \wedge z = f(i)).$$

$\pi_1(i)$	$\ldots$	$\pi_k(i)$
$z := h_1(i)$	$\ldots$	$z := h_k(i)$

8.3. Rekurzív függvény

Legyen H egy tetszőleges halmaz, $k > 0$ egy egész szám, továbbá $f : \mathcal{Z} \to H, h : \mathcal{Z} \times H^k \to T$ olyan függvények, hogy $f(0) = t_0, f(-1) = t_{-1}, .., f(-k+1) = t_{-k+1}$ és $\forall i > 0 : f(i+1) = h(i+1, f(i), ..., f(i-k+1))$. Feladatunk az, hogy meghatározzuk az f függvény $n > 0$ helyen felvett értékét.

$A = \mathcal{Z} \times H, \quad n : \mathcal{Z}, \quad z : H.$

$B = \mathcal{Z}, \quad n' : \mathcal{Z}.$

$Q_{n'} = (n = n' \wedge n > 0),$

$R_{n'} = (Q_{n'} \wedge z = f(n)).$

$P_{n'} = (Q_{n'} \wedge 0 \le i \le n \wedge z = f(i), z_{-1} = f(i-1), .., z_{-k+1} = f(i-k+1)).$

<table>
<tr><td colspan="2">$i, z, z_{-1}, .., z_{-k+1} := 0, t_0, t_{-1}, .., t_{-k+1}$</td></tr>
<tr><td colspan="2">$i \ne n$</td></tr>
<tr><td rowspan="2"></td><td>$z, z_{-1}, .., z_{-k+1} := h(i+1, f(i), ..., f(i-k+1)), z, .., z_{-k+2}$</td></tr>
<tr><td>$i := i + 1$</td></tr>
</table>

8.4. Elemenkénti feldolgozás

Jelölések: Legyen H tetszőleges halmaz.

$$X \overset{\text{def}}{=} X_1 \times ... \times X_n, X_i \subseteq 2^H (i = 1, ..., n),$$

$$Y \overset{\text{def}}{=} Y_1 \times ... \times Y_m, Y_i \subseteq 2^H (i = 1, ..., m).$$

$$x, \overline{x}, \overline{\overline{x}} \in X.$$

1. Def.: *(Teljesen diszjunkt felbontás)* Azt mondjuk, hogy $\overline{x}, \overline{\overline{x}}$ teljesen diszjunkt felbontása x -nek, ha $\forall i \in [1, n] : x_i = \overline{x}_i \cup \overline{\overline{x}}_i$ és $\forall i, j \in [1, n] : \overline{x}_i \cap \overline{\overline{x}}_j = \emptyset$.
2. Def.: *(Elemenként feldolgozható függvény)* Legyen $f : X \to Y$. Ha minden $x \in X$ minden $\overline{x}, \overline{\overline{x}}$ teljesen diszjunkt felbontására
   i) $f(\overline{x}) \cup f(\overline{\overline{x}}) = f(x)$,
   ii) $f(\overline{x}) \cap f(\overline{\overline{x}}) = \emptyset$,
   akkor f -et elemenként feldolgozhatónak nevezzük.

$i.$) Első változat

Legyen $f : X \to Y, X, Y \subseteq 2^H$ függvény. $\boxed{29}$

$A = A_1 \times A_2,$

$A_1 = set(H_1), \ A_2 = set(H_2), \ H_1 \subseteq H, H_2 \subseteq H, \quad x : A_1, z : A_2.$

$B = A_1, \quad x' : A_1.$

$Q_{x'} = (x = x'),$

$R_{x'} = (z = f(x')).$

$P_{x'} = (z \cup f(x) = f(x') \wedge z \cap f(x) = \emptyset).$

$z := \emptyset$
$f(x) \neq \emptyset$
$e : mem(x)$
$z := z \tilde{\cup} f(\{e\})$
$x := x \tilde{-} \{e\}$

ii.) **Második változat**

Legyen az f kétváltozós függvény. $\boxed{30}$

$$A = A_1 \times A_2 \times A_3, \quad A_i = set(H_i), H_i \subseteq H (i = 1, 2, 3), \quad x : A_1, y : A_2, z : A_3.$$
$$B = A_2 \times A_3, \quad x' : A_1, y' : A_2.$$
$$Q_{x',y'} = (x = x' \wedge y = y'),$$
$$R_{x',y'} = (z = f(x', y')).$$
$$P_{x',y'} = (z \cup f(x, y) = f(x', y') \wedge f(x, y) \cap z = \emptyset$$
$$\wedge (x' \setminus x) \cap y = \emptyset \wedge (y' \setminus y) \cap x = \emptyset).$$

$z := \emptyset$		
$f(x, y) \neq \emptyset$		
$e : mem(x \cup y)$		
$e \in x \wedge e \notin y$	$e \notin x \wedge e \in y$	$e \in x \wedge e \in y$
$z := z \tilde{\cup} f(\{e\}, \emptyset)$ $x := x \tilde{-} \{e\}$	$z := z \tilde{\cup} f(\emptyset, \{e\})$ $y := y \tilde{-} \{e\}$	$z := z \tilde{\cup} f(\{e\}, \{e\})$ $x := x \tilde{-} \{e\}$ $y := y \tilde{-} \{e\}$

iii.) **Harmadik változat**

Az f függvényről feltesszük, hogy (f_1, f_2) alakú. $\boxed{31}$

$$A = A_1 \times A_2 \times A_3, \quad A_i = set(H_i \subseteq H)(i = 1, 2, 3), \quad x : A_1, y : A_2, z : A_3.$$
$$B = A_1, \quad x' : A_1.$$
$$Q_{x'} = (x = x'),$$
$$R_{x'} = (y = f_1(x') \wedge z = f_2(x')).$$
$$P_{x'} = (y \cup f_1(x) = f_1(x') \wedge y \cap f_1(x) = \emptyset \wedge z \cup f_2(x) = f_2(x') \wedge z \cap f_2(x) = \emptyset).$$

$y, z := \emptyset, \emptyset$
$f_1(x) \neq \emptyset \vee f_2(x) \neq \emptyset$
$e : mem(x)$
$y, z := y \tilde{\cup} f_1(\{e\}), z \tilde{\cup} f_2(\{e\})$
$x := x \tilde{-} \{e\}$

$iv.)$ **Általános változat**

Rögzített m -re, n -re: $z_i = f_i(x_1, ..., x_n)(i = 1, ..., m)$. $\boxed{32}$

$$P_{x'_1,...,x'_n} = (\forall i \in [1, m] : (z_i \cup f_i(x_1, ..., x_n) = f_i(x'_1, ..., x'_n)$$
$$\wedge z_i \cap f_i(x_1, ..., x_n) = \emptyset) \wedge (\forall i, j \in [1, n] : (x'_i \setminus x_i) \cap x_j = \emptyset)).$$

	$y_1, ..., y_m := \emptyset, ..., \emptyset$	
	$\cup_{i=1}^{m} f_i(x_1, ..., x_n) \neq \emptyset$	
	$e : mem(\cup_{i=1}^{n} x_i)$	
$\setminus \dots$	$\setminus$ $e \in x_{i_1} \wedge \dots e \in x_{i_k} \wedge e \notin x_{i_{k+1}} \wedge e \notin x_{i_n}$	$\setminus \dots$
$\dots$	$y_j := y_j \tilde{\cup} f_j(sl(\{i_1 \dots i_k\}, e)_j)(j \in \{i_1, ..., i_k\})$ $x_{i_1}, \dots, x_{i_k} := x_{i_1} \tilde{-} \{e\}, \dots, x_{i_k} \tilde{-} \{e\}$	$\dots$

ahol $i_1, ..., i_n$ az $1, ..., n$ permutációja, $sl : 2^{\{1,...,n\}} \times H \to Y$,

$$sl(\{i_1, .., i_k\}, e)_i = \begin{cases} \{e\}, & \text{ha } i \in \{i_1, ..., i_k\} \\ \emptyset, & \text{ha } i \notin \{i_1, ..., i_k\} \end{cases}$$

az elágazás "ágainak" száma $2^n - 1$,

MÁSODIK RÉSZ

PROGRAMTRANSZFORMÁCIÓK

Definíciók

Legyen $A = A_1 \times ... \times A_n$, $a_i : A_i$, $S \subseteq A \times A^{**}$.

1. Azt mondjuk, hogy az S program $p(S)$ programfüggvénye független az A_i állapottérkomponenstől, ha $\forall a, b \in \mathcal{D}_{p(S)} : (\forall k \in ([1, i-1] \cup [i+1, n]) : a_k = b_k) \rightarrow p(S)(a) = p(S)(b)$.
2. Azt mondjuk, hogy az $a_i : A_i$ változó konstans függvény az S programban, ha $\forall a \in A : \forall \alpha \in S(a) : (\forall \alpha_k \in \alpha : \alpha_{k\,i} = a_i)$.
3. Azt mondjuk, hogy az S program végrehajtása nem változtatja meg az $a_i : A_i$ változót, ha $\forall a \in \mathcal{D}_{p(S)} : p(S)(a)_i = a_i)$.

Tételek

1. Nem megengedett elágazásfeltételek kitranszformálása

Legyen $S \subseteq A \times A^{**}$, ahol $S = IF(\pi_1 : S_1, ..., \pi_n : S_n)$, $A = A_1 \times ... \times A_m$. Legyen $S_{IF} \subseteq A' \times A'^{**}$, ahol $A' = A_1 \times ... \times A_m \times \mathcal{L}_1 \times ... \times \mathcal{L}_n$, $a_i : A_i, l_j : \mathcal{L}_j$.

$S' \subseteq A' \times A'^{**}$
$l_1 := \pi_1(a_1, ..., a_m)$
...
$l_n := \pi_n(a_1, ..., a_m)$

Ekkor az $S_{IF} \stackrel{\text{def}}{=} (S'; IF(l_1 : S_1, ..., l_n : S_n)$ program ekvivalens A-n az S programmal.

2. Nem megengedett ciklusfeltétel kitranszformálása

Legyen $S \subseteq A \times A^{**}$, ahol $S = DO(\pi, S_0)$, $A = A_1 \times ... \times A_m$. Legyen $S_{DO} \subseteq A' \times A'^{**}$, ahol $A' = A_1 \times ... \times A_m \times \mathcal{L}$, $a_i : A_i, l : \mathcal{L}$.

$S_{DO} \subseteq A' \times A'^{**}$	
$l := \pi(a_1, ..., a_m)$	
l	
	S_0
	$l := \pi(a_1, ..., a_m)$

Ekkor az S_{DO} program ekvivalens A-n az S programmal.

3. Szimultán értékadás kitranszformálása

$S \subseteq A \times A^{**}$, ahol $A = A_1 \times ... \times A_m$ $S' \subseteq A' \times A'^{**}$, ahol
$A' = A_{11} \times ... \times A_{m1} \times A_{12} \times ... \times A_{m2}$, $a_{i1} : A_{i1}$, $a_{i2} : A_{i2}$.
$S \stackrel{\text{def}}{=} a_{11}, ..., a_{m1} := f_1(a_{11}, ..., a_{m1}), ..., f_m(a_{11}, ..., a_{m1})$.

$S' \subseteq A' \times A'^{**}$
$a_{12}, ..., a_{m2} := f_1(a_{11}, ..., a_{m1}), ..., f_m(a_{11}, ..., a_{m1})$
$a_{11}, ..., a_{m1} := a_{12}, ..., a_{m2}$

Ekkor az S' program ekvivalens A-n az S programmal.

Megj.: Speciális esetek, amikor f_i nem függ az A minden komponensétől.

4. Szekvencia két tagjának felcserélése

Ha $S_{12} = (S_1; S_2), S_{21} = (S_2; S_1)$ és minden olyan állapottérkomponenst, amelytől S_1 nem függelen, az S_2 végrehajtása nem változtat meg és viszont, akkor S_{12} ekvivalens S_{21}-gyel az A-n.

5. Ciklusmagban levő szekvencia két tagjának felcserélése:

Ha $S = DO(\pi, S_0)$, $S_0 = (i := i + 1; S_{01})$, $S' = ({S_{01}}^{i \leftarrow i+1}; i := i + 1)$ és i konstans függvény S_{01}-ben, akkor S ekvivalens S'-vel az A-n.

6. Függvény helyettesítése változóval

Legyen f az állapottér $A_{i_1} \times ... \times A_{i_k}$ alterén értelmezett függvény, melynek értékkészlete a H halmaz ($f \circ (a_1, ..., a_n)$ az A-n értelmezett függvény). Ha S-ben az $a_{i_1}, ..., a_{i_k}$ változók konstans függvények, akkor S ekvivalens az

$S' = (z := f(a_{i_1}, ..., a_{i_k}); S^{f(a_{i_1}, ..., a_{i_k}) \leftarrow z})$ programmal az A-n, ahol

$S' \subseteq A' \times A'^{**}$ és $A' = A_1 \times ... \times A_n \times H$, $a_i : A_i$, $z : H$.

7. Rekurzív függvény kitranszformálása

Legyen H egy tetszőleges halmaz, $k > 0$ egy egész szám, továbbá $f : \mathcal{Z} \to H$, $h : \mathcal{Z} \times H^k \to T$ olyan függvények, hogy $f(0) = t_0$, $f(-1) = t_{-1}$,..., $f(-k+1) = t_{-k+1}$ és $\forall i > 0 : f(i + 1) = h(i + 1, f(i), ..., f(i - k + 1))$.

S	
$i := 0$	
S_1	
π	
	S_0
	$i := i + 1$

ahol i konstans függvény úgy S_1-ben, mint S_0-ban, valamint S_0-ban hivatkozás történik $f(i + 1)$-re. Ekkor S ekvivalens a következő programmal az A-n:

$i, z, z_{-1}, .., z_{-k+1} := 0, t_0, t_{-1}, .., t_{-k+1}$
S_1
π
$z, z_{-1}, .., z_{-k+1} := h(i+1, f(i), ..., f(i-k+1)), z, .., z_{-k+2}$
${S_0}^{f(i+1) \leftarrow z}$
$i := i + 1$

8. Programinverzió

...

9. Kiemelési szabályok

a.) $IF(\pi_1 : (S; S_1), \pi_2 : (S; S_2))$ ekvivalens A-n az $S; IF(\pi_1 : S_1, \pi_2 : S_2$ programmal, ha a π_1 és π_2 függvényekre és az S programra alkalmazható a függvény helyettesítése változóval transzformáció.

b.) $IF(\pi : a :=\uparrow, \neg\pi : a :=\downarrow) \equiv a := \pi$.

...

TÍPUSTRANSZFORMÁCIÓK PROGRAMOZÁSI TÉTELEKRE

1. Sorozatról file-ra előreolvasási technikával

Legyen T egy tetszőleges típus és legyen $a : seq(T; low, lorem, dom)$. Legyen továbbá $x : seq(T; lopop)$ egy olyan file, amelynek végét egy extremális elem jelzi.

	$dx, x : lopop$
$a.low$	dx
$a : lorem$	$dx, x : lopop$
$a.dom \neq 0$	$dx \neq EXTR$

2. Halmazról sorozatra elemenkénti feldolgozással

Legyen T tetszőleges típus és legyen $x, y, z : set(T)$. Legyenek továbbá $a, b : seq(T; dom, low, lorem)$ szigorúan monoton növekvő sorozatok, és $c : seq(T; hiext)$ egy tetszőleges sorozat.

$z := \emptyset$	$c := ()$
$x \neq \emptyset \vee y \neq \emptyset$	$a.dom \neq 0 \vee b.dom \neq 0$
$e : mem(x \cup y)$	$e := min(a.low, b.low)$
$z \tilde{\cup}$	$c : hiext$
$x \tilde{-}$	$a : lorem$
$e \in x$, $e \notin x$	$e = a.low$, $e < a.low$!!

3. Intervallumon értelmezett függvényről sorozatra

Legyenek m és n tetszőleges egész számok ($m \leq n$), legyen T tetszőleges típus és legyen $f : fun([m,n],T)$. Legyen továbbá $x : seq(T; dom, low, lorem$ egy sorozat.

$i = n$
$f(i+1)$
$i := i + 1$
$f(m)$

$x.dom = 0$
$x.low$
$x : lorem$
$x.low$

4. Sorozatról vektorra (vagy függvény típusra)

Legyen T tetszőleges típus és legyen $a : seq(T; dom, low, lorem, hiext)$. Legyen továbbá $v : Vektor(\mathcal{Z}, T)$ egy vektor ($i, j \in [v.lob, v.hib], i \leq j$).

$a.dom = 0$
$a.low$
$a : lorem$
$a : hiext(e)$

$i = j$
$v(i)$
$i := i + 1$
$v(j+1), j := e, j + 1$

ÁLLAPOTTÉRTRANSZFORMÁCIÓ

...

ADATABSZTRAKCIÓ

...

FÜGGVÉNYABSZTRAKCIÓ

...

VISSZALÉPÉSES KERESÉS

1. A feladat megfogalmazása

Legyenek $U_1, \ldots, U_n$ véges halmazok, ahol az egyes halmazok számosságát jelölje σ_i, azaz $\sigma_i \stackrel{\text{def}}{=} |U_i|$ ($\forall i \in [1,n]$) és tegyük fel, hogy $\sigma_1 \neq 0$. Jelölje U az U_i halmazok direktszorzatát: $U \stackrel{\mathrm{def}}{=} \times_{i=1}^{n} U_i$. Legyen $\varrho : U \to \mathcal{L}$ egy olyan tulajdonság (logikai függvény), amelyhez megadható egy ϱ_i $(i \in [0, n])$ tulajdonságsorozat, amelyre:

$$
\begin{aligned}
&i)\ \ \varrho_0 \stackrel{\mathrm{def}}{=} igaz,\\
&ii)\ \ \varrho_{i+1}(u) \Rightarrow \varrho_i(u) \quad (\forall i \in [0, n-1]),\\
&iii)\ \forall j \in [1, i] : u_j = v_l \ \ \Rightarrow \ \ \varrho_i(u) = \varrho_i(v),\\
&\qquad azaz\ \varrho_i\ csak\ az\ U\ első\ i\ komponensétől\ függ,\\
&iv)\ \ \varrho_n = \varrho.
\end{aligned}
$$

A feladat: döntsük el, hogy van-e olyan $u \in U$, amelyre $\varrho(u)$ igaz, s ha van, adjunk is meg egy ϱ tulajdonságú elemet!

2. A feladat specifikációja

Legyen $N = \times_{i=1}^{n} V_i$, ahol $V_i = [0, \sigma_i - 1] \subset \mathcal{N}_0$ $(\forall i \in [1, n])$. Ekkor $|N| = |U|$. Sorszámozzuk meg az U_i-beli elemeket 0-tól $\sigma_i - 1$ -ig $(\forall i \in [1, n])$, s jelölje $u_{i_j} \in U_i$ a megfelelő elemeket. Ekkor kölcsönösen egyértelműen megfeleltethetjük egymásnak N és U elemeit — jelölje ezt a leképezést ϕ. Így $\phi : N \to U$, s ha $\nu \in N$, akkor $\phi(\nu) = (u_{1_{\nu_1}}, \ldots, u_{n_{\nu_n}})$. Az N elemei tekinthetők vegyes számrendszerbeli számoknak. Ezzel definiálunk ui. egy rendezést N -en, így van értelme egy elem "rákövetkezőjéről" beszélni.

Jelöljük $f(\nu)$ -vel a $\nu \in N$ vegyes számrendszerbeli szám értékét, azaz legyen

$$f(\nu) \stackrel{\mathrm{def}}{=} \sum_{i=1}^{n} (\nu_i \prod_{j=i+1}^{n} \sigma_j).$$

Ha ν', $\nu'' \in N$, akkor $\nu' < \nu'' \Leftrightarrow f(\nu') < f(\nu'')$. Jelölje továbbá

$$
\begin{aligned}
&\varepsilon_0 \text{ az } N-\text{beli zéruselemet} : \varepsilon_0 \stackrel{\mathrm{def}}{=} (0, 0, \ldots, 0),\\
&\varepsilon_n \text{ az } N-\text{beli egységelemet} : \varepsilon_n \stackrel{\mathrm{def}}{=} (0, 0, \ldots, 1),\\
&\varepsilon_i \stackrel{\mathrm{def}}{=} (0, 0, \ldots, i, \ldots, 1) \text{ azt az } N-\text{beli értéket, amelyre } f(\varepsilon_i) \stackrel{\mathrm{def}}{=} \prod_{j=i+1}^{n} \sigma_j.
\end{aligned}
$$

A fentiek figyelembevételével megfogalmazhatjuk a feladat specifikációját:

$$
\begin{aligned}
&A = N \times \mathcal{L}, \qquad \nu : N, \qquad l : \mathcal{L}.\\
&Q = (igaz),\\
&R = (l = (\exists \nu' \in N : \varrho(\phi(\nu'))) \wedge l \Rightarrow \varrho(\phi(\nu)).
\end{aligned}
$$

3. A megoldás előállítása

A fenti feladat — ha nem használjuk ki a ϱ speciális jellemzőit — visszavezethető a lineáris keresés tételének harmadik változatára a következő megfeleltetésekkel:

$$\begin{array}{lcl} [m,n] & \longrightarrow & [f^{-1}(0), f^{-1}(|N|-1)] \\ u & \longrightarrow & l \\ \gamma & \longrightarrow & \varrho \\ \delta & \longrightarrow & f(\nu) = |N|-1 \end{array}$$

A kapott program:

$l, v, \nu := \varrho(\phi(\varepsilon_0)), (f(\varepsilon_0) = \|N\| - 1), \varepsilon_0$
$\neg l \wedge \neg v$
$\nu := \nu \oplus \varepsilon_n$
$l := \varrho(\phi(\nu))$
$v := (f(\nu) = \|N\| - 1)$

Lényegesen növelhetjük az algoritmus hatékonyságát, ha kihasználjuk a ϱ speciális tulajdonságát, azaz azt, hogy ha $\varrho_i(\phi(\nu))$ igaz és $\varrho_{i+1}(\phi(\nu))$ hamis, akkor minden olyan ν'-re, amely az első $i+1$ komponensében megegyezik ν-vel, $\varrho_{i+1}(\phi(\nu'))$ hamis lesz, a ϱ $iii)$ tulajdonsága miatt. Ekkor tehát a ν-vel együtt elvethetjük az összes ilyen ν'-t is!

Így célszerűbb $\nu \oplus \varepsilon_n$ helyett $\nu \oplus \varepsilon_{i+1}$ -et venni, ami tulajdonképpen a ν vegyes számrendszerbeli szám inkrementálása az $(i+1)$. pozíción. Az inkrementálást könnyebben kezelhetjük a programban, ha ν-t kiegészítjük egy túlcsordulást jelző bittel — jelölje ezt c. Ekkor, ha c értéke 1 -re változik, ez azt jelenti, hogy nincs több lehetőségünk értelmes ν választására.

A fentiekből következik, hogy az $l := \varrho(\phi(\nu))$ értékadást érdemes kiegészíteni annak vizsgálatával, hogy melyik az a legkisebb index, amelyre már $\varrho_i(\phi(\nu))$ hamis.

Alkalmazzuk a levezetési szabályokat! A ciklus invariánsának válasszuk a következőt:

$$\begin{aligned} P : & (\forall \nu', (0 \leq f(\nu') < f(\nu) + c|N| : \neg\varrho(\phi(\nu')))\wedge \\ & l = \varrho(\phi(\nu)) \wedge \neg l \Rightarrow (\varrho_{m-1}(\phi(\nu)) \wedge \neg\varrho(\phi(\nu)) \wedge (\forall i \in [m+1, n] : \nu_i = 0)). \end{aligned}$$

Ha ciklusfeltételnek $\pi = \neg l \wedge (c = 0)$ -t választjuk, akkor $P \wedge \neg\pi \Rightarrow R$ valóban teljesül. A terminátor függvény legyen $t = |N| - f(\nu) - c|N|$. Amíg $P \wedge \pi$ igaz, addig a $t > 0$ nyilván teljesül. A t csökkentéséről a ν inkrementálásával gondoskodunk.

$\nu, c, m := \varepsilon_0, 0, 1$
$KER(\nu, m, l)$
$\neg l \wedge (c = 0)$
$SUM(\nu, m, c)$
$KER(\nu, m, l)$

A $KER(\nu, m, l)$ programnak biztosítania kell a ciklus $Q \Rightarrow P$ feltételének teljesülését is.

$$
\begin{aligned}
&A_{KER} = N \times \mathcal{N}_0 \times \mathcal{L}, \qquad \nu : N, \qquad m : \mathcal{N}_0, \qquad l : \mathcal{L}.\\
&B_{KER} = N \times \mathcal{N}_0, \qquad \nu' : N, \qquad m' : \mathcal{N}_0.\\
&Q_{KER} = (\nu = \nu' \wedge m = m' \wedge \varrho_{m'-1}(\phi(\nu))),\\
&R_{KER} = (\nu = \nu' \wedge \varrho_{m'-1}(\phi(\nu)) \wedge l = (\forall i \in [m', n] : \varrho_i(\phi(\nu)))\\
&\qquad \wedge \neg l \Rightarrow (\neg \varrho_m(\phi(\nu)) \wedge \forall i \in [m', m-1] : \varrho_i(\phi(\nu)))).
\end{aligned}
$$

Ez a feladat visszavezethető a lineáris keresés harmadik változatára, azzal a megfontolással, hogy az az állítás, hogy egy tulajdonság egy intervallum minden elemére igaz, ekvivalens azzal az állítással, hogy nem létezik az intervallumnak olyan eleme, amelyre a tulajdonság nem igaz.

$$
\begin{aligned}
u &\longrightarrow \neg l\\
i &\longrightarrow m\\
\gamma &\longrightarrow \varrho(m)\\
\delta &\longrightarrow (m = n)
\end{aligned}
$$

$KER(\nu, m, l)$

$l, m := \uparrow, m - 1$

$l \wedge (m \neq n)$

$l := \varrho_{m+1}(\phi(\nu))$

$m := m + 1$

A főprogramban így $Q' = (R_{KER} \wedge (\nu = \varepsilon_0) \wedge (c = 0))$ lesz, s ebből $Q' \Rightarrow P$ belátható.

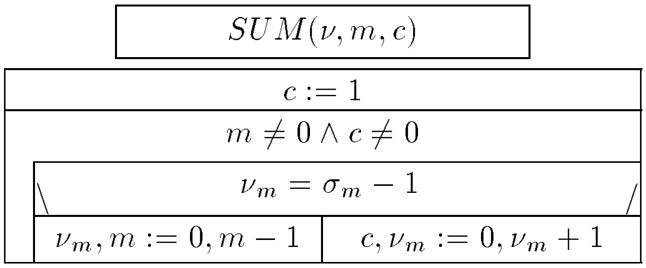
A $SUM(\nu, m, c)$ program legyen a következő:

$$
\begin{aligned}
&A_{SUM} = N \times \mathcal{N}_0 \times \mathcal{E}, \qquad \nu : N, \qquad m : \mathcal{N}_0, \qquad c : \mathcal{E}.\\
&B_{SUM} = N \times \mathcal{N}_0, \qquad \nu' : N, \qquad m' : \mathcal{N}_0.\\
&Q_{SUM} = (\nu = \nu' \wedge m = m'),\\
&R_{SUM} = ((f(\nu) + c|N| = f(\nu') + \prod_{i=m'+1}^{n} \sigma_i) \ \wedge \ m \in [0, m'] \ \wedge\\
&\qquad (\forall i \in [m+1, m'] : \nu_i = 0) \wedge (c = 0 \Rightarrow \nu_m \neq 0)).
\end{aligned}
$$

Invariáns tulajdonságnak a következőt vehetjük:

$$
\begin{aligned}
&P_{SUM} : ((f(\nu) + c \prod_{i=m+1}^{n} \sigma_i = f(\nu') + \prod_{i=m'+1}^{n} \sigma_i) \wedge m \in [0, m'] \wedge\\
&\qquad (\forall i \in [m+1, m'] : \nu_i = 0) \wedge (c = 0 \Rightarrow \nu_m \neq 0)).
\end{aligned}
$$

A ciklusfeltétel legyen $\pi_{SUM} = (m \neq 0) \wedge (c \neq 0)$, s ekkor belátható, hogy $P_{SUM} \wedge \neg\pi_{SUM} \Rightarrow R_{SUM}$. A terminátorfüggvény legyen $t_{SUM} = m + c$. Amíg $P_{SUM} \wedge \pi_{SUM}$ igaz, addig $t_{SUM} > 0$ is nyilván teljesül. A t_{SUM} értéke akkor csökken, ha vagy m-et, vagy c-t csökkentjük.



Legyen $Q'_{SUM} = (\nu = \nu' \wedge m = m' \wedge c = 1)$, ekkor $Q'_{SUM} \Rightarrow P_{SUM}$ teljesül. $P_{SUM} \wedge \pi \Rightarrow lf(S_{0_{SUM}}, P_{SUM})$ belátásához igazolnunk kell, hogy

$$i)\ P_{SUM} \wedge \pi \wedge \nu_m = \sigma_m - 1 \Rightarrow lf(\nu_m := 0; m := m-1, P_{SUM})$$
$$ii)\ P_{SUM} \wedge \pi \wedge \nu_m = \sigma_m - 1 \Rightarrow lf(c := 0; \nu_m := \nu_m + 1, P_{SUM})$$

teljesül. Ezek beláthatók $f(\nu)$ definíciójának segítségével.

A SUM program levezetése után a főprogram helyességéhez már csak a $P \wedge \pi \Rightarrow lf(SUM; KER, P)$ teljesülése szükséges, s ez a fentiekből következik.

IDŐSZERŰSÍTÉS

1. Az időszerűsítés definíciója

Tegyük fel, hogy van egy adatállományunk, ami azonos típusú rekordok sorozata. Nevezzük ezt törzsfile-nak:

$$T = it(E),$$
$$to : T.$$

Továbbá adott transzformációk egy sorozata, nevezzük ezt módosítófile-nak:

$$M = it(F), F = \{\, f \mid f\ :\ T\ \rightarrow\ T\, reláció \,\},$$
$$m : M.$$

A feladat a törzsfile aktualizálása a módosítófile alapján, azaz a módosítófile-beli transzformációsorozat végrehajtásával kapott új törzsfile meghatározása. Jelölje ezt a transzformációt:

$$t = id(to, m), \quad másképpen \quad t = (\, m_{|m|} \circ \ldots \circ m_1)(to).$$

Ez a megfogalmazás még túl általános, a függvény tényleges kiszámíthatóságához néhány további tulajdonságát is meg kell adnunk.

i.) Legyenek *to* és *m* kulcsos szekvenciális file-ok.

$$T = seq(E), \quad E = (K, D)$$
$$M = seq(F), \quad F = (K, V)$$

ahol K tetszőleges rendezett halmaz, a rekord kulcsrésze, D az adatrész típusa, és V adja a végrehajtandó transzformációt.
ii.) Legyen *to* kulcs szerint egyértelmű file, azaz egy kulcs csak egyszer forduljon elő benne.

$$to : T, \quad \forall i, j \in [1, to.dom] : t_i.K = t_j.K \Leftrightarrow i = j.$$

iii.) Végül leszűkítjük a lehetséges transzformációkat is:

$$V = (W1; W2; W2),$$
$$W1 = \{törlés\},$$
$$W2 = (\{beszúrás\}, D),$$
$$W3 = (\{javitás\}, M'),$$
$$M' = seq(G), \qquad ahol\ G - be\ a\ D\ \rightarrow\ D\ alakú\ leképezések\ tartoznak.$$

Definiáljuk ezen módosító transzformációk hatását a törzsfile-ra! (A definíciókat halmazokra írjuk fel, ami *to* kulcs szerinti egyértelműsége miatt megengedhető.) Legyen $m_i : M$.
Az $m_i.W1$, azaz a *törlés* transzformáció hatása

$$\{m_i(t)\} = \{t_j | t_j \in t \wedge t_j.K \neq m_i.K\}.$$

Az $m_i.W2$, azaz a *beszúrás* transzformáció hatása

$$\{m_i(t)\} = \{t\} \cup \begin{cases} \emptyset & , \text{ha } \exists k \in [1, t.dom] : t_k.K = m_i.K \\ \{(m_i.K, m_i.D)\} & , \text{ha } \exists k \in [1, t.dom] : t_k.K = m_i.K \end{cases}$$

Az $m_i.W3$, azaz a *módositás* transzformáció hatása

$$\{m_i(t)\} = \{t_j | t_j \in t \wedge t_j.K \neq m_i.K\} \cup$$
$$\cup \begin{cases} \emptyset & , \text{ha } \exists k \in [1, t.dom] : t_k.K = m_i.K \\ \{(m_i.K, m_i.M'(t_k.D)) & \\ \quad | t_k \in t \wedge m_i.K = t_k.K\} & , \text{ha } \exists k \in [1, t.dom] : t_k.K = m_i.K \end{cases}$$

Def.: *(Közönséges időszerűsítés)* Az időszerűsítést közönségesnek nevezzük, ha az *i.*), *ii.*) és *iii.*) megkötések teljesülnek rá.

Def.: *(Egyszerű időszerűsítés)* A közönséges időszerűsítést egyszerűnek nevezzük, ha G egyértelmű és konstans transzformáció. Ekkor egy javítás egyetlen konstans transzformáció, vagyis valójában csere.

2. Egyértelmű módosítófile esete

A következőkben egyértelmű módosító file esetén három különböző mélységű absztrakción alapuló megközelítéssel állítunk elő egy-egy megoldást.

Tegyük fel tehát, hogy a módosítófile egyértelmű kulcs szerint, s adjuk meg a feladat specifikációját:

$$
\begin{aligned}
&A = T \times T \times M, \quad t : T, \quad to : T, \quad m : M. \\
&B = T \times M, \quad to' : T, \quad m' : M. \\
&Q = (to = to' \wedge m = m' \wedge \ \textit{az időszerűsités egyszerű és } m' \textit{ egyértelmű}), \\
&R = (t = id(to', m')).
\end{aligned}
$$

2.1. Visszavezetés halmazok uniójára

Milyen kulcsok lesznek az új törzsfile-ban, t-ben? Biztos, hogy csak olyanok, amelyek vagy to'-ben, vagy m'-ben, vagy mindkettőben szerepeltek. Próbáljuk meg visszavezetni a feladatot két halmaz uniójára!

Jelölje $K(to)$ a to-beli, $K(m)$ az m-beli kulcsok halmazát, valamint $k(to)$ azt a to-beli, és $k(m)$ azt az m-beli elemet, aminek a kulcsa k.

Bővítsük az adatok halmazát: $D' = (D \cup \{üres\})$.

Az egyes transzformációk értelmezési tartománya és értékkészlete felírható D' segítségével:

$$
\begin{aligned}
&W1 : D \to \{üres\} && -- \textit{ a törlés} \\
&W2 : \{üres\} \to D && -- \textit{ a beszúrás} \\
&W3 : D \to D && -- \textit{ a módositás}
\end{aligned}
$$

Ha egy elem adatrésze $üres$, akkor ez az elem nem létezik, nem írhatjuk ki az új törzsfile-ba.

Emlékeztetőül a $z = x \cup y$ program:

$z := \emptyset$		
$x \neq \emptyset \vee y \neq \emptyset$		
$e : mem(x \cup y)$		
$e \in x \wedge e \notin y$	$e \notin x \wedge e \in y$	$e \in x \wedge e \in y$
$z := z \tilde{\cup} \{e\}$ $x := x \tilde{-} \{e\}$	$z := z \tilde{\cup} \{e\}$ $y := y \tilde{-} \{e\}$	$z := z \tilde{\cup} \{e\}$ $x := x \tilde{-} \{e\}$ $y := y \tilde{-} \{e\}$

Mi felel meg ennek az időszerűsítési feladatban?

$$x \qquad \longrightarrow \qquad to$$

$$
\begin{array}{lcl}
y & \longrightarrow & m \\
z & \longrightarrow & t \\
x \neq \emptyset \vee y \neq \emptyset & \longrightarrow & to \neq \emptyset \vee m \neq \emptyset \\
e : mem(x \cup y) & \longrightarrow & k : mem(K(to) \cup K(m)) \\
e \in x & \longrightarrow & k \in K(to) \\
e \in y & \longrightarrow & k \in K(m)
\end{array}
$$

A megfelelő program:

$t := \emptyset$			
$to \neq \emptyset \vee m \neq \emptyset$			
	$k : mem(K(to) \cup K(m))$		
	$k \in K(to) \wedge k \notin K(m)$	$k \notin K(to) \wedge k \in K(m)$	$k \in K(to) \wedge k \in K(m)$
	$S1$	$S2$	$S3$

Vizsgáljuk az elágazás egyes ágainak megfelelő programokat!

Első eset. Ha a k kulcs csak to-ban szerepelt, m-ben nem, az azt jelenti, hogy nincs erre az elemre vonatkozó módosítás m-ben. Ekkor hozzá kell venni az új törzsfile-hoz, és el kell hagyni to-ból.

$S1$
$t := t \cup \{k(to)\}$
$to := to - \{k(to)\}$

Második eset. Ha a k kulcs csak m-ben szerepelt, to-ban nem, akkor csak a *beszúrás* transzformáció hajtható végre helyesen, s az m-ből kell elhagyni a k kulcsú transzformációt.

$S2$

$k(m).W1$	$k(m).W2$	$k(m).W3$
$hiba(1)$	$t := t \cup \{(k, k(m).D)\}$	$hiba(2)$
$m := m - \{k(m)\}$		

Harmadik eset. Ha a k kulcs to-ban is és m-ben is szerepelt, akkor a *törlés* és a *javítás* transzformációk hajthatók végre, s mindkét file-ból el kell hagyni a k kulcsú elemet:

$S3$		
$k(m).W1$	$k(m).W2$	$k(m).W3$
$SKIP$	$hiba(3)$ $t := t \cup \{k(to)\}$	$t := t \cup \{(k, k(m).G(k(to).D))\}$
$to := to - \{k(to)\}$ $m := m - \{k(m)\}$		

Ha a program hibajelzéseket is ad, azokat a jelzett ágakon kell megvalósítani.

Térjünk vissza az eredeti feladathoz, amikor a törzsfile-t és a módosítófile-t nem halmazként, hanem szekvenciális file-ként kezeljük. Tegyük fel, hogy mindkét file elemei kulcs szerint rendezettek, s a file végét egy extremális (minden más kulcsnál nagyobb kulcsú) elem jelöli. A megoldó program ekkor:

$t := \emptyset$ $dto, to : lopop$ $dm, m : lopop$		
$dto.K \neq EXTR \vee dm.K \neq EXTR$		
$k : mem(K(to) \cup K(m))$		
$dto.K < dm.K$	$dm.K < dto.K$	$dm.K = dto.K$
$t : hiext(dto)$ $dto, to : lopop$	$S2$	$S3$
$t : hiext(EXTR)$		

$S2$		
$dm.W1$	$dm.W2$	$dm.W3$
$hiba(1)$	$t : hiext(dm.K, dm.D)$	$hiba(2)$
$dm, m : lopop$		

$S3$		
$dm.W1$	$dm.W2$	$dm.W3$
$SKIP$	$hiba(3)$ $t : hiext(dto)$	$t : hiext(dto.K, dm.D)$
$dto, to : lopop$ $dm, m : lopop$		

2.2. Megoldás az elemenkénti feldolgozás fogalmának segítségével

A halmazműveletek — mint a fenti unió — általánosításával, az absztrakció szintjét növelve, eljuthatunk az elemenkénti feldolgozás fogalmához, s így megfogalmazhatjuk, mi az, ami ezekben a feladatokban közös, ami miatt az elemenként feldolgozható függvények helyettesítési értékét kiszámító program mindig ugyanaz.

Az előbbi feladat megoldását kétféleképpen is visszavezethetjük az elemenkénti feldolgozásra.

2.2.1. Visszavezetés egyváltozós elemenkénti feldolgozásra

Az időszerűsítési feladat állapotterét elképzelhetjük úgy is, hogy ha vesszük a törzsfile-beli és a módosítófile-beli kulcsok halmazának unióját, lesznek olyanok, amelyekhez csak egy törzsfile-beli adatrész tartozik, lesznek olyanok, amelyek a törzsfileban nem szerepelnek, s végül, lesznek olyanok, amelyek mindkettőben. Így gondolatban a törzsfile-ból és a módosítófile-ból létrehozhatunk egy olyan állományt, amelyek elemei három komponensből állnak: egy kulcsból (vagy to-ból, vagy m-ből), egy to-beli adatrészből (aminek értéke esetleg $üres$), és egy m-beli transzformációrészből (ami esetleg $üres$, s ez az identikus leképezést jelzi). A két utóbbi mező természetesen nem lehet egyszerre $üres$. Formálisan leírva:

$$
\begin{aligned}
&X = seq(DX),\\
&DX = (K, D', V'),\\
&D' = D \cup \{üres\},\\
&V' = V \cup \{üres\}.
\end{aligned}
$$

Legyen $x : X$. Ekkor $K(x) = K(to) \cup K(m)$. Jelölje dx az x egy tetszőleges elemét. Ekkor az előbbiek alapján

$$
dx.D' = \begin{cases} \text{üres} & \text{, ha } dx.K \notin K(to) \\ dx.K(to).D & \text{, ha } dx.K \in K(to) \end{cases}
$$

$$
dx.V' = \begin{cases} \text{üres} & \text{, ha } dx.K \notin K(m) \\ dx.K(m).V & \text{, ha } dx.K \in K(m) \end{cases}
$$

Állapottértranszformációt alkalmazunk, így a feladat ezen az állapottéren felfogható egyváltozós elemenkénti feldolgozásnak, a következő f függvénnyel: $f(\{e\}) = (e.K, e.V'(e.D))$, azaz a transzformációs részt alkalmazzuk az adatrészre

$$
\begin{aligned}
&e.V'.üres(e.D') = e.D'\\
&e.V.W1(e.D') = üres,\ ha\ e.D' = üres\\
&e.V.W2(üres) = e.V.W2.D\\
&e.V.W3(e.D') = e.V.W3.D,\ ha\ e.D' = üres
\end{aligned}
$$

A specifikáció ezen az állapottéren:

$$\begin{aligned}
&A = X \times T, \qquad x : X, \qquad t : T.\\
&B = X, \qquad x' : X.\\
&Q = (x = x' \wedge \ az\ időszerűsités\ egyszerű),\\
&R = (t = f(x')).
\end{aligned}$$

A program:

$t := \emptyset$
$x \neq \emptyset$
$e : mem(x)$
$d := f(\{e\})$
$t := \cup\{d\}$
$x := x - \{e\}$

A $d := f(\{e\})$ program megvalósítása:

$\boxed{d := f(\{e\})}$

$e.D' \neq üres \wedge e.V' = üres$	$e.D' = üres \wedge e.V' \neq üres$	$e.D' \neq üres \wedge e.V' \neq üres$
$d :\ (e.K, e.D')$	$hiba(3)$ $t : hiext(dto)$	$t : hiext(dto.K, dm.D)$

Térjünk vissza az eredeti állapottérre:

$$\begin{array}{lcl}
x \neq \emptyset & \longrightarrow & to \neq \emptyset \vee m \neq \emptyset\\
e : mem(x) & \longrightarrow & k : mem(K(to) \cup K(m))\\
e.D' \neq üres \wedge e.V' = üres & \longrightarrow & k \in K(to) \wedge k \notin K(m)\\
e.D' = üres \wedge e.V' \neq üres & \longrightarrow & k \notin K(to) \wedge k \in K(m)\\
e.D' \neq üres \wedge e.V' \neq üres & \longrightarrow & k \in K(to) \wedge k \in K(m)\\
t := t \cup \{d\} & \longrightarrow & t,\ ha\ d = \emptyset\\
 & & t \cup \{d\},\ ha\ d \neq \emptyset\\
x := x - \{e\} & \longrightarrow & to := to - \{k(to)\},\ ha\ k \in K(to) \wedge k \notin K(m)\\
 & & m := m - \{k(m)\},\ ha\ k \notin K(to) \wedge k \in K(m)\\
 & & to := to - \{k(to)\}\\
 & & m := m - \{k(m)\},\ ha\ k \in K(to) \wedge k \in K(m)
\end{array}$$

A $t := t \cup \{d\}$ és az $x := x - \{e\}$ programok függetlenek egymástól, így felcserélhetők. Felhasználva, hogy a $d := f(\{e\})$ és az $x := x - \{e\}$ utasításokat megvalósító programok feltételrendszere megegyezik, valamint beépítve az uniót a $d := \emptyset$-t nem tartalmazó ágakba, pontosan azt a programot kapjuk, mint az előbb.

2.2.2. Visszavezetés kétváltozós elemenkénti feldolgozásra

Feladatunkat talán legegyszerűbben a kétváltozós egyértékű elemenkénti feldolgozásra vezethetjük vissza (ill. a kétváltozós kétértékű elemenkénti feldolgozásra, ha a hibajelzéseket is kezelni akarjuk):

$$\begin{aligned}
&A = T \times M \times T, \qquad to : T, \qquad m : M, \qquad t : T.\\
&B = T \times M, \qquad to' : T, \qquad m' : M.\\
&Q = (to = to' \wedge \ az\ időszerűsités\ egyszerű\ és\ m'\ egyértelmű),\\
&R = (t = id(to', m')).
\end{aligned}$$

Az időszerűsítő függvény (a speciális feltételek miatt kulcs szerint elemenként feldolgozható) a következőképpen adható meg:

$$\begin{aligned}
id(k, \emptyset) &= k(to)\\
id(\emptyset, k) &= \begin{cases} \emptyset & \text{, ha } k(m).W1\\ (k, k(m).D) & \text{, ha } k(m).W2\\ \emptyset & \text{, ha } k(m).W3 \end{cases}\\
id(k, k) &= \begin{cases} \emptyset & \text{, ha } k(m).W1\\ k(to) & \text{, ha } k(m).W2\\ (k, k(m).D) & \text{, ha } k(m).W3 \end{cases}
\end{aligned}$$

A feladatot megoldó program triviális behelyettesítéssel adódik.

3. Nem egyértelmű módosítófile esete

Abban az esetben, ha nem tételezhetjük fel, hogy az m módosító file egyértelmű, a feladat már nem lesz elemenként feldolgozható.

Ekkor két megoldási absztrakció kínálkozik (továbbra is kihasználjuk természetesen, hogy állományaink kulcs szerint rendezettek).

3.1. Adatabsztrakciós megközelítés

Állapottértranszformációt alkalmazunk, a módosítófile-t úgy tekintjük, hogy elemei $(kulcs, transzformációsorozat)$ típusúak, azaz:

$$M' = seq(F'),\ ahol\ F' = (K, V''),\ ahol\ V'' = seq(V).$$

Ekkor, ezzel a módosítófile-lal, újból elemenkénti feldolgozáshoz jutunk, csak a megfelelő transzformációsorozatot kell majd megvalósítani az eredeti állapottéren.

$$\begin{aligned}
id(k, \emptyset) &= k(to)\\
id(\emptyset, k) &= (k, k(m).V'')\\
id(k, k) &= (k, k(m).V''(k(to).D))
\end{aligned}$$

(A transzformációk kompozíciója az egyes transzformációk értelmezési tartományai és értékkészletei miatt csak a transzformációk megfelelő sorozatára értelmes.)

A kapott program:

<table>
<tr><td colspan="4">$t := \emptyset$
$dto, to : lopop$
$dm, m : lopop$</td></tr>
<tr><td colspan="4">$dto.K \neq EXTR \vee dm.K \neq EXTR$</td></tr>
<tr><td rowspan="2"></td><td>$dto.K < dm.K$</td><td>$dm.K < dto.K$</td><td>$dm.K = dto.K$</td></tr>
<tr><td>$t : hiext(dto)$
$dto, to : lopop$</td><td>$TRANSZF(üres)$</td><td>$TRANSZF(dto.D)$
$dto, to : lopop$</td></tr>
<tr><td colspan="4">$t : hiext(EXTR)$</td></tr>
</table>

$TRANSZF(PAR)$

<table>
<tr><td colspan="7">$d := PAR$
$ak := dm.K$</td></tr>
<tr><td colspan="7">$ak = dm.K$</td></tr>
<tr><td rowspan="4"></td><td colspan="2">$dm.W1$</td><td colspan="2">$dm.W2$</td><td colspan="2">$dm.W3$</td></tr>
<tr><td colspan="2">$d = üres$</td><td colspan="2">$d = üres$</td><td colspan="2">$d = üres$</td></tr>
<tr><td>$hiba(1)$</td><td>$d := üres$</td><td>$d := dm.D$</td><td>$hiba(2)$</td><td>$hiba(3)$</td><td>$d := dm.D$</td></tr>
<tr><td colspan="6">$dm, m : lopop$</td></tr>
<tr><td colspan="7">$d = üres$</td></tr>
<tr><td colspan="3">$SKIP$</td><td colspan="4">$t : hiext(EXTR)$</td></tr>
</table>

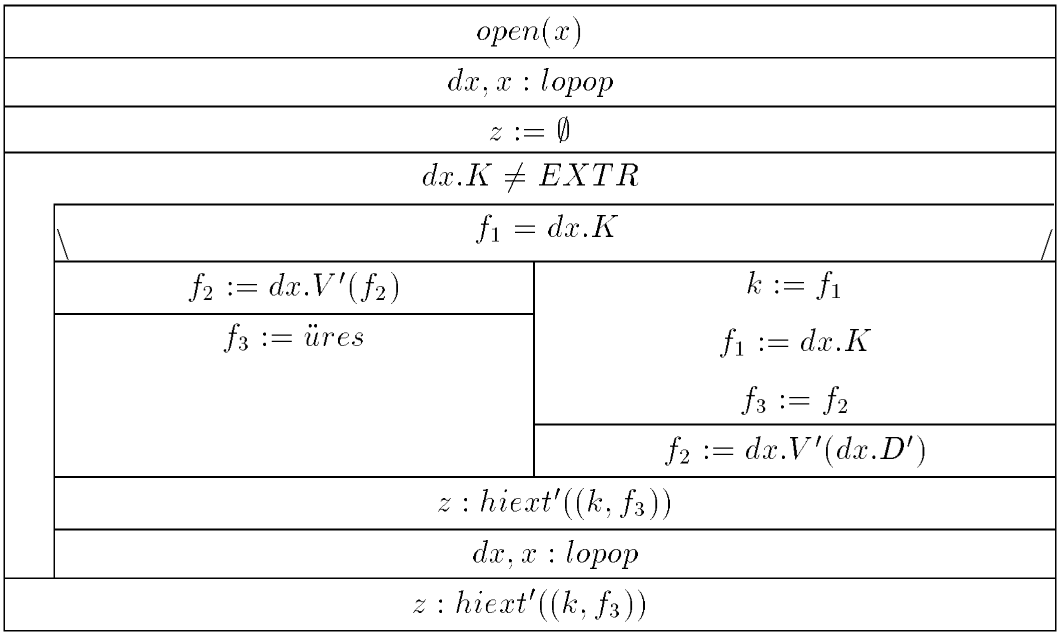
3.2. Függvénytranszformációs módszer

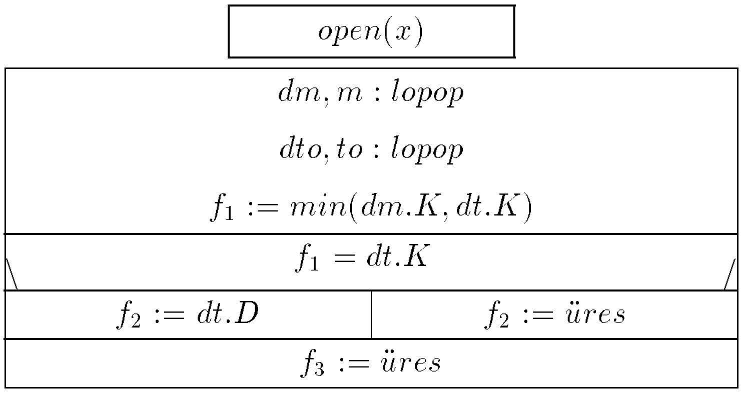
A feladat megoldását általában önkéntelenül befolyásolja a specifikációja is. A fenti egyváltozós elemenkénti feldolgozásból kiindulva nem egyértelmű módosítófile esetén definiálhatunk egy rekurzívan megadott függvényt, amely a megfelelő transzformációsorozatot kiértékeli. A megoldás ebben a megközelítésben (X definícióját ld. fent, azzal a különbséggel, hogy most kulcs szerint nem egyértelmű):

$$A = X \times T, \qquad x : X, \qquad t : T.$$
$$B = X, \qquad x' : X.$$
$$Q = (x = x' \wedge \; x \; kulcs \; szerint \; rendezett),$$
$$R = (t = CONC_{i=1}^{x'.dom}(f(i)_1, f(i+1)_3).$$

ahol $CONC$ csak olyan adatokat konkatenál, amire $f_3 \neq üres$. Az f függvény definíciója:

$$f : bbfZ \to K \times D' \times D'$$
$$f(0) = (min(x.low.K, EXTR), x.low.D', \ddot{u}res)$$
$$f(i+1) = \begin{cases} (f(i)_1, x_{i+1}.V'(f)i)_2), \ddot{u}res) & \text{, ha } f(i)_1 = x_{i+1}.K \\ (x_{i+1}.K, x_{i+1}.V'(x_{i+1}.D'), f(i)_2) & \text{, ha } f(i)_1 \neq x_{i+1}.K \end{cases}$$





$dx, x : lopop$

$dm.K = EXTR \wedge dt.K = EXTR$			
$dx.K := EXTR$	$dt.K < dm.K$	$dt.K > dm.K$	$dt.K = dm.K$
	$dx.K := dt.K$ $dx.D' := dt.D$ $dx.V' := üres$ $dto, to : lopop$	$dx.K := dm.K$ $dx.D' := üres$ $dx.V' := dm.V$ $dm, m : lopop$	$dx.K := dm.K$ $dx.D' := dt.D$ $dx.V' := dm.V$ $dto, to : lopop$ $dm, m : lopop$

$f_2 := dx.V'(p)$

$dx.V' = üres$						
$f_2 := p$	$dx.V'.W1$		$dx.V'.W2$		$dx.V'.W3$	
	$p = üres$		$p = üres$		$p = üres$	
	$hiba$	$f_2 := üres$	$f_2 := dx.D$	$hiba$	$hiba$	$f_2 := dx.V(p)$

$z : hiext'((k, f_3))$

$f_3 = üres$	
$SKIP$	$z : hiext((k, f_3))$

LISTÁZÁS

...

HARMADIK RÉSZ

ALGORITMUSOK ÉS ADATSZERKEZETEK

Gráfalgoritmusok

...

String keresése stringben

(Pl. Knuth-Morris-Pratt módszer, Boyer-Moore módszer, Dömölki módszere.)
...

Kódolás, tömörítés

(Pl. Huffman-kódolás.)
...

PROGRAMOZÁSI TECHNIKA

A modul fogalma

...

Szövegkezelés

...

File-kezelés

...

Menükezelés

...

KIEGÉSZÍTÉSEK, RÉSZLETES MAGYARÁZATOK

1 Bevezetés

Ez a jegyzet programozó matematikusok programozástanulásának bevezetéséhez készült. Ebből következik, hogy benne a programozás matematikai szemléletű megközelítésére törekszünk.

Az első dolgunk logikusan az lenne, hogy megmondjuk, mi is az a programozás. Ez azonban nem is olyan egyszerű feladat. Miért? Próbáljunk egy egyszerű meghatározást adni. ”A feladatot felbontjuk a gép számára már érthető utasítások sorozatára”, vagy ... stb.

Mindenképpen szerepel a meghatározásokban két szó: feladat és gép. De mi az, hogy feladat? Valójában nincs az életünknek olyan területe, problémája ami nem tekinthető ebben az értelemben feladatnak. Ahhoz, hogy értelme legyen a fogalomnak, meg kell keresni, hogy mi a közös a számtalan sok különböző feladatban.

És mit jelent az a szó, hogy gép? Tudjuk, hogy nagyon sok különböző számítógép van. Ezekhez rendkívül sokféle kiegészítő berendezés tartozhat. Sőt, több — esetleg nagyon sok — ”egyszerű” gép kapcsolódhat egymáshoz. Ezen kívül emberi tevékenységek, döntések is befolyásolják a gépek, a géprendszerek működését. Vajon ők nem részei a feladatot megoldó gépnek?

Lehet-e egyáltalán programozásról beszélni ilyen heterogén feltételek mellett? És mi lesz tíz vagy húsz év múlva?

Mi a megoldás? Az absztrakció és a szétválasztás.

A programozás során nem a konkrét feladat és a gépi lehetőségek között teremtünk közvetlen kapcsolatot, hanem választunk egy absztrakt modellt, amiben megfelelő pontossággal leírhatjuk a konkrét feladatot. Ennek az a haszna, hogy az absztrakt modell matematikai eszközökkel kezelhető, a programozási ismeretek pontosan megfogalmazhatók. A probléma szétválik a konkrét feladat modellezésére, az absztrakt programozási feladatra, és a gépi megvalósításra (az implementációra).

A későbbiekben még sokszor látjuk majd, hogy ez a két eszköz — az absztrakció és a szétválasztás — milyen alapvető a programozási tevékenység minden területén.

Még egy kérdést meg kell vizsgálnunk. Ki programozó? Ez sem triviális kérdés. Már régóta nagyon sokan ”programoztak”. Gondoljunk arra, hogy a logarléc is számológép, és annak használata is lényegében programozás volt, ugyanúgy, mint a jelenleg használt asztali és egyéb kalkulátorok, gépek használata.

Vajon mi különbözteti meg a programozót a programkészítőtől? Akinek pl. autója van, az valamilyen szinten képes az autót gondozni, apróbb hibáit kijavítani. Erre ”képesítése” is van, hiszen a jogosítvány megszerzésekor vizsgázott műszaki ismeretekből. Mégis világos, hogy nem mindenki autószerelő szakember, aki autót szerel. Miben különbözik az autószerelő az átlagos autót szerelőtől?

A két kérdésre hasonló a válasz.

A programozó felkészültsége, gyakorlata miatt, és mert bonyolultabb eszközökkel (nem gépi, hanem logikai, matematikai eszközökkel) rendelkezik — és azokat használni is tudja —, ezért képes bonyolultabb feladatok megoldására.

Ahogy egy amatőrnek lehetnek remek műszerei, és profi szakembereket felülmúló ügyessége, érzéke az autószereléshez, ugyanúgy elképzelhető, hogy valaki képesítés szerint nem, de tudása alapján programozó.

Ez azonban nem jelenti azt, hogy aki képesítése szerint programozó, annak nem volna kötelessége megfelelő ismeretekkel rendelkezni.

Ebben a jegyzetben csak az absztrakt programozási feladatokkal foglalkozunk. Ismertetünk egy lehetséges eszköztárat ezek tudományos igényű kezeléséhez.

2 Állapottér

Ha a konkrét valóság valamilyen részét modellezni akarjuk, el kell dönteni, hogy mik azok a jellemzők, amiket a modellben figyelembe veszünk, és ezek milyen értékeket vehetnek föl. Egy-egy jellemző lehetséges értékeinek halmazát jelöltük a definícióban A_{i_j} -vel, és neveztük típusértékhalmaznak. Ez az elnevezés azt fejezi ki, hogy az A_{i_j} halmaz hasonló, azonos tulajdonságú dolgok lehetséges értékeinek halmaza. Itt most nem foglalkozunk azzal, hogy a hasonlóság miben áll.

Mivel különböző jellemzők lehetséges értékeinek halmaza lehet azonos, ezért a definícióban az A_{i_j}-k nem feltétlenül különbözők.

Az A_{i_j} halmazokról feltehettük, hogy legfeljebb megszámlálhatóak, hiszen más halmazok úgysem kezelhetők számítástechnikai eszközökkel.

Az, hogy egy adott probléma modellezésénél mi lesz az állapottér, nem egyértelmű. Például egy pont térbeli helyezetét jellemezhetjük derékszögű- vagy polárkoordinátákkal is. Egy sík pontjai megadhatók kétdimenziós térben, de háromdimenziósban is.

Az állapottér megválasztása nagyon fontos dolog, hiszen alapvetően meghatározza, hogy mit és hogyan tudunk leírni. Ha az állapottér túl kevés komponensből áll, akkor van, amit nem tudunk leírni benne, s ha túl sok, akkor fölöslegesen bonyolult lesz a modell. Az, hogy túl sok, vagy túl kevés a komponens, relatív fogalom, attól függ, hogy milyen kérdések vizsgálatára használjuk a modellt.

Általában egy bonyolult, összetett probléma esetén először a legáltalánosabb, legmagasabb szintű kérdéseket vizsgáljuk. Ekkor a modell durva, csak a legfontosabb jellemzőket tartalmazza. Az egyre pontosabb, részletesebb vizsgálatokhoz fokozatosan finomítjuk a modellt is, az állapotteret új komponensekkel bővítjük.

Előfordulhat az is, hogy egy-egy részletkérdést mint önálló problémát az állapottér egy alterén vizsgálunk.

(Megjegyzés. Az állapotteret elképzelhetjük úgy is, mint egy gép (adott program által használt) memóriája, regiszterei, adatai és eredményei lehetséges értékeinek halmazát. Ilyenkor az állapottér egy pontját felfoghatjuk az értékek egy sorozatának, amely egy adott időpontban a programozó számára minden fontos információt tartalmaz a gép állapotáról.)

3 A változó fogalma

A változók segítségével egyszerűsíthetjük az állapottéren értelmezett állítások, relációk, tehát feladatok, programfüggvények, predikátumok, leggyengébb előfeltételek stb. leírását.

Mivel minden változó értelmezési tartománya az állapottér, és értékkészlete egy típusértékhalmaz, egy változót jellemezhetünk egy típussal is — így beszélhetünk a változó típusáról. Azt, hogy egy x változó értékkészlete T, így jelöljük: $x : T$.

A jelölések egyszerűsége érdekében, ha egy formulában minden változó azonos argumentummal szerepel, azt seholsem írjuk ki. Ha $x : T$ egy változó, f egy tetszőleges T-n értelmezett függvény, az $f(x)$ jelölést $f(x(a))$ (ahol $a \in A$) rövidítéseként használjuk. Legyen például $A = \mathcal{Z} \times \mathcal{Z}$ és $x, y : \mathcal{Z}$. Az $R = (x > 0 \wedge y > 0)$ jelölés az $(x(a) > 0 \wedge y(a) > 0 \wedge a \in A)$ állítást jelöli, és $[R] = \{(a_1, a_2) \in A | a_1 > 0 \wedge a_2 > 0\}$. Ha π predikátum, a $\pi = (x > y)$ azt jelenti, hogy $\pi(a) = \begin{cases} igaz & , ha\ x(a) > y(a) \\ hamis & , ha\ x(a) \leq y(a) \end{cases}$.

A változók segítségével könnyen felírhatunk olyan függvényeket, amik az állapottér bizonyos komponensein értelmezett $v_1, \ldots, v_n$ változók függvényei, $f \circ (v_1, \ldots, v_n)$ alakban. Ha nem okoz félreértést, ezt így is írjuk: $f(v_1, \ldots, v_n)$.

4 A feladat fogalma

A feladat definíciója azt fejezi ki, hogy minden programozási feladat megfogalmazható mint egy leképezés az állapottér bizonyos pontjaiból az állapottérbe. Ez a leképezés általában nem egyértelmű.

Lássunk egy példát. A feladat legyen az, hogy keressük meg egy természetes számnak egy természetes osztóját. Mi lehet az állapottér? Mivel két dologról van szó, két számról, ezért kézenfekvő lehetőség, hogy állapottérnek az $A = \mathcal{N} \times \mathcal{N}$ halmazt válasszuk. Az első komponens azoknak a lehetséges értékeknek a halmaza, amiknek az osztóját keressük, a második pedig azoké, amik előfordulhatnak mint osztók. Ekkor a feladat az, hogy olyan $(x, y) \in A$ pontba jussunk, ahol $y|x$. Pontosabban egy $(u, v) \in A$ pontból egy $(x, y) \in A$ pontba kell jutni, ahol $y|x$ és $u = x$. Tehát a feladat

$$F = \{((u, v), (x, y)) \in A \times A |\ x = u \wedge y|x\}.$$

A konkrét feladat megfogalmazása olyan, hogy más absztrakt feladathoz is

eljuthatunk:
$$F' = \{((u,v),(x,y)) \in A \times A \mid \; y|u\}.$$

Világos, hogy F és F' különböző feladatok. Ez rámutat az absztrakt modell készítésének egyik fontos okára is: nagyon gyakori, hogy a modellezés közben azért jutunk különböző absztrakt feladatokhoz, mert a konkrét feladat nem pontos, gyakran csak a "fejünkben van meg", és ezért lényeges hiányosságok lehetnek benne.

Az absztrakt modell elkészítése után átgondoljuk ismételten, hogy abban vajon valóban az igazi céljainkat fogalmaztuk-e meg.

A következő példánk nem egy tetszőleges, hanem a legnagyobb valódi osztó megkeresése.

Az állapottér maradhat az, ami az előző feladatban volt, a feladat pedig a következő G vagy G' reláció:

$$\begin{aligned} G = \{((u,v),(x,y)) \in A \times A| \; & u \textit{ összetett szám}, \; x = u, \\ & y \textit{ az } x \textit{ legnagyobb valódi osztója}\}, \\ G = \{((u,v),(x,y)) \in A \times A| \; & u \textit{ összetett szám}, \\ & y \textit{ az } u \textit{ legnagyobb valódi osztója}\}. \end{aligned}$$

Most a G reláció az összetett számokon értelmezett függvény.

Felmerülhet a kérdés, hogy miért nem az összetett számok halmazát választottuk az állapottér egyik komponensének. Erre a kérdésre később, a típus tárgyalásakor még visszatérünk, egyelőre az "egyszerűbb" halmazok használata mellett döntünk.

(Megjegyzés. Szokásos a programozási feladatot input-output modellként is felfogni: milyen adatokból milyen eredményt kell gyártani. Sokszor azonban — különösen bonyolultabb programrendszerek esetében — nehéz megmondani, mik az adatok és mik az eredmények. Ugyanakkor nyilvánvaló, hogy egy input-output szemléletben megfogalmazott feladat minden nehézség nélkül felírható a definíciónak megfelelő reláció formájában.)

5 A program fogalma

A konkrét programok egyik legszembetűnőbb tulajdonsága a dinamikus viselkedés, a működés. Itt azonban a program definíciója "statikus", nem a működés, hanem a működés eredményének absztrakciója. Ebben az a programozói tapasztalat is tükröződik, hogy a "mit csinál" a fontos, és nem a "hogyan". Másrészt ebben a statikus modellben is értelmezhetők az elterjedt dinamikus fogalmak megfelelői. Pédául:

a.) Az $a \in \mathcal{R}_S$ sorozatot az S program egy végrehajtásának mondjuk.

b.) Az S program az $a \in A$ pontból kiindulva a $b \in A$ pontban terminál, ha $\exists \alpha \in S(a) \cap A^* : b = \tau(\alpha)$.

c.) Az S program az $a \in A$ pontból indítva terminál, ha $S(a) \subset A^*$.

d.) S egy mindenütt termináló program, ha $\forall a \in A : S(a) \subset A^*$.

e.) Az S program az $a \in A$ pontból indítva a $b \in A$ pontban abortál, ha $\exists \alpha \in S(a) \cap A^\infty \Rightarrow \exists i \in \mathcal{N} : (\forall j (j > i) : \alpha_j = b)$.

f.) Ha a feladat (program, programfüggvény) függvény, akkor a feladatot (programot, programfüggvényt) determinisztikusnak, ellenkező esetben nemdeterminisztikusnak nevezzük.

(Megjegyzés. Az abortálásra is be lehetne vezetni a terminálásnak megfelelően az a pontból indítva abortáló, és a mindenütt abortáló program fogalmát.)

A definíció szerint a programot az egész állapottéren értelmeztük. Ez egyrészt megfelel a valóságnak — egy program valamit minden esetben csinál —, másrészt fölösleges lenne az értelmezési tartomány korlátozását itt lehetővé tenni, hiszen erre szolgál pl. a programfüggvény fogalma. Az a.) kikötés lényegében az üres sorozatot zárja ki a program értékkészletéből.

A program normális működését az állapottéren történő mozgásnak tekintjük. A sorozatok ezt a mozgást rögzítik. Ezért zártuk ki, hogy ilyenkor a sorozatokban egymás mellett szerepelhessen az állapottérnek ugyanaz a pontja. Ezt a lehetőséget, az ismétlődést, az abnormális működés jelzésére használjuk. Ilyenkor a program viselkedése nem írható le a modell keretein belül, s ilyen esetben a további működésről semmit sem tudunk mondani.

(Megjegyzés. A megállás gépi szintű realizációja általában egy ugró utasítás saját magára. Azaz a program működik, de nem jut sehova, külső segítségre vár.)

$\boxed{6}$ A programfüggvény fogalma

A programozót elsősorban az érdekli, hogy egy program egy kezdeti állapotból elindítva mit csinál, azaz terminál-e (véges-e a sorozat), és ha igen, hova, milyen állapotba jutott (mi a sorozat utolsó eleme). Ezért vezettük be a programfüggvény fogalmát.

A programfüggvény nem a program működését, hanem csak a "jól működés" eredményét adja meg. A "nem rosszul működés" eredményét is megadhatjuk, erre szolgál a szabad programfüggvény ($p'(S)$).

$$\begin{aligned}
&\mathcal{D}_{p'(S)} = \{a \in A | S(a) \cap A^* \neq \emptyset\},\\
&\forall a \in \mathcal{D}_{p'(S)} :\\
&\quad p'(S)(a) = \{b \in A | \exists \alpha \in S(a) \cap A^* \ : \ \tau(\alpha) = b\}.
\end{aligned}$$

Nyilvánvalóan $\mathcal{D}_{p(S)} \subset \mathcal{D}_{p'(S)}$.

A programfüggvényt nemcsak "gyengébbre", hanem "erősebbre" is választhatjuk. A korlátos programfüggvény jele $p_k(S)$.

$$\begin{aligned}
&\mathcal{D}_{p_k(S)} = \{a \in A | S(a) \subset A \wedge \ S(a) \ \textit{véges halmaz}\},\\
&\forall a \in \mathcal{D}_{p_k(S)} :\\
&\quad p_k(S)(a) = \{b \in A | \exists \alpha \in S(a) \ : \ \tau(\alpha) = b\}.
\end{aligned}$$

Nyilván $\mathcal{D}_{p_k(S)} \subset \mathcal{D}_{p(S)}$.

A korlátos programfüggvény nemcsak azt garantálja, hogy a program terminál, hanem azt is, hogy korlátos idő alatt.

7 A megoldás fogalma

A megoldás definíciója azt a két kézenfekvő követelményt tartalmazza, hogy a) a program bármelyik lehetséges kezdeti állapotból elindítva biztosan termináljon, és b) a végállapot feleljen meg a feladat követelményeinek. Ha egy programot az állapottérnek egy rögzített pontjából többször elindítunk, előfordulhat, hogy különböző pontokban terminál.

A definíció b.) kikötése pontosan azt jelenti, hogy ezeknek a pontoknak mindegyike a feladatnak megfelelő legyen. Tehát egy feladat szempontjából nem a program, hanem csak a programfüggvény érdekes. Mivel az azonos programfüggvényű programok azonos hatásúaknak tekinthetők, ezért ezeket ekvivalens programoknak nevezzük.

(Megjegyzés. Valójában az ekvivalens programok egyéb tulajdonságai nagyon különbözők lehetnek, ami nem elhanyagolható. Ilyen elsősorban a hatékonyság: milyen nagyságú gépet mennyi ideig vesz igénybe a program. Ha készítünk egy programot, és az megoldása a feladatnak, azt mondjuk, hogy a program helyes. Azonban ez még nem jelenti azt, hogy jó munkát végeztünk, hogy jó a program. Pl. egy helyes program lehet teljesen használhatatlan, mert olyan gépi kapacitásigénye van, ami sokszorosa egy vele ekvivalens másik programnak. A program "jósága" nagyon sok dologtól függ, ennek tárgyalásával itt még nem foglalkozhatunk.)

8 A kiterjesztésre vonatkozó tételek

A tételek állításainak igazolása előtt bebizonyítjuk, hogy minden program ekvivalens a kiterjesztettjével.

<u>*Lemma*</u>

Legyen B altere A-nak, S egy program B-n, S' az S kiterjesztettje A-n. Ekkor S és S' ekvivalensek B-n.

<u>*Bizonyitás*</u>

Azt kell belátnunk, hogy $pr_{B\times B}(p(S')) = pr_{B\times B}(p(S)) = p(S)$.

a.) $pr_B(\mathcal{D}_{p(S')}) = \mathcal{D}_{p(S)}$.
A programfüggvények értelmezési tartományainak projekciója megegyezik. Ez abból következik, hogy a projekció nem változtatja meg egy sorozat hosszát.

b.) Legyen $a \in \mathcal{D}_{p(S')}, b \in \mathcal{D}_{p(S)}$ úgy, hogy $pr_B(a) = b$ teljesül. Ilyen a és b létezik (ha $\mathcal{D}_{p(S)} \neq \emptyset$) a fentiek szerint. Belátjuk, hogy $pr_B(p(S')(a)) = p(S)(b)$. Mivel $a \in \mathcal{D}_{p(S')}$, ezért tetszőleges $\alpha \in S(a)$ véges, azaz létezik $\tau(\alpha)$. $pr_B(p(S')(a)) = pr_B(\cup_{\alpha\in S'(a)}\tau(\alpha)) = \cup_{\alpha\in S'(a)}pr_B(\tau(\alpha)) = \cup_{\alpha\in S'(a)}\tau(pr_B(\alpha)) = p(S)(pr_B(a)) = p(S)(b)$. $\bowtie$

<u>*Az 1.) állítás igazolása*</u>

Az állítás két összefüggés belátását kívánja.

a.) $\mathcal{D}_G \subset \mathcal{D}_{p(S)}$ igazolása.
Ez abból következik, hogy a feltételeink szerint a hasonló $\mathcal{D}_G \subset \mathcal{D}_{p(S')}$ fennáll, valamint abból, hogy $pr_B(\mathcal{D}_{p(S')}) = \mathcal{D}_{p(S)}$ és $pr_B(\mathcal{D}_G) = \mathcal{D}_G$.

b.) Tetszőleges $b \in \mathcal{D}_{G_1}$-re $p(S)(b) \subset G_1(b)$ kell, hogy teljesüljön. Ha $b \in \mathcal{D}_{G_1}$, akkor $\exists a \in \mathcal{D}_G$, amelyre $pr_B(a) = b$. A feltételek szerint S' megoldása G-nek, így $a \in \mathcal{D}_{p(S')}$ és $p(S')(a) \subset G(a)$. A lemmából következik, hogy $pr_B(p(S')(a)) = p(S)(pr_B(a)) = p(S)(b)$. Ezért $p(S)(b) \subset pr_B(G(a)) \subset G_1(b)$. $\bowtie$

A 2.) állítás igazolása

Az állítás két összefüggés belátását kívánja.

a.) $\mathcal{D}_F \subset \mathcal{D}_{p(S)}$ igazolása.
Mivel S_1 megoldása F'-nek, fennáll $\mathcal{D}_{F'} \subset \mathcal{D}_{p(S_1)}$. S_1 és S ekvivalensek B-n, ezért $pr_B(\mathcal{D}_{p(S_1)}) = \mathcal{D}_{p(S)}$. F' az F feladat kiterjesztése, így $pr_B(\mathcal{D}_{F'}) = \mathcal{D}_F$. A felírt három összefüggésből már következik a bizonyítandó állítás.

b.) Be kell látni, hogy $\forall b \in \mathcal{D}_F : p(S)(b) \subset F(b)$. Írjuk fel, hogy mit jelent ebben a speciális esetben S_1 és S ekvivalenciája. Tetszőlegesen rögzített $b \in \mathcal{D}_F$ esetén tekintsük mindazon $a \in A$ pontokat, amelyekre $pr_B(a) = b$. Ekkor $p(S)(b) = \cup_{pr_B(a)=b} pr_B(p(S_1)(a))$. Mivel S_1 megoldása F'-nek ezért $\forall a \in \mathcal{D}_{F'} : p(S_1)(a) \subset F'(a)$. Ezt felhasználva $p(S)(b) = \cup_{pr_B(a)=b} pr_B(p(S)(a)) \subset \cup_{pr_B(a)=b} pr_B(F'(a)) = \cup_{pr_B(a)=b} F(pr_B(a)) = F(b)$. $\bowtie$

A 3.) állítás igazolása

A lemma szerint S' ekvivalens S-sel B-n. Másrészt nyilván $pr_{B\times B}(F') = F$ (F' éppen a legbővebb ilyen feladat). Ekkor alkalmazható egyik előző állításunk, mely szerint, ha S' megoldása F'-nek, akkor S megoldása F-nek.

a.) $\mathcal{D}_{F'} \subset \mathcal{D}_{p(S)}$ igazolása.
Nyilván $\mathcal{D}_{p(S')} = \{a \in A | pr_B(a) \in \mathcal{D}_{p(S)}\}$ és $\mathcal{D}_{F'} = \{a \in A | pr_B(a) \in \mathcal{D}_F\}$. A feltétel szerint $\mathcal{D}_F \subset \mathcal{D}_{p(S)}$. Ezekből az összefüggésekből az állítás (pl. indirekt módon) könnyen belátható.

b.) Tekintsünk tetszőleges $a \in \mathcal{D}_{F'}$-t. Legyen $b = pr_B(a)$. Definíciónk alapján $p(S')(a) = \{a' \in A | pr_B(a') \in p(S)(b) \wedge pr_B(a') = pr_B(a)\}$, $F'(a) = \{a' \in A | pr_B(a') \in F(b)\}$. Mivel $p(S)(b) \subset F(b)$, ezért az előzőkhöz hasonló módon adódik, hogy $p(S')(a) \subset F'(a)$. Mivel $p(S')(a)$ definíciójában a'-ra még további megszorítás is van, ezért általában $p(S')(a) \neq F'(a)$. $\bowtie$

9 Leggyengébb előfeltétel

A leggyengébb előfeltétel Dijkstra-tól származó eredeti megfogalmazása: "The condition that characterizes the set of *all* initial stets such that activation will certainly result in a properly terminating happening leaving the system in a final state satisfying a given post-condition is called 'the weakest pre-condition corresponding to that post-condition'". Ennek a fogalomnak sokan sokféle definícióját fogalmazták meg (Wand, Guerreiro, Roever, Plotkin, Hehner stb.), illetve foglalkoztak ezek kapcsolataival. Mi most az eredeti definíció megfelelőjét fogalmaztuk meg a mi modellünk segíségével.

A leggyengébb előfeltétel kijelöli azt a legtágabb halmazt, aminek elemeire az S program biztosan terminál, és a végállapotra igaz R. Fontos észrevenni, hogy $[lf(S,R)]$-en kívül is lehetnek olyan pontjai az állapottérnek, amikből a program $[R]$-ben juthat, csak azokból nem biztos, hogy oda jut.

Definiálhatunk az $lf(S,R)$-nél gyengébb feltételt is, ami csak azt biztosítja, hogy ha a program valahova jut, azaz terminál, akkor ez csak $[R]$-ben lehet. Ezt a feltételt leggyengébb szabad előfeltételnek nevezzük:

$$[lf'(S,R)] ::= \{a \in \mathcal{D}_{p'(S)} | p'(S)(a) \subset [R]\},$$

ahol $p'(S)$ az szabad programfüggvény.

A leggyengébb korlátos előfeltételt a $p_k(S)$ korlátos programfüggvény segítségével definiáljuk.

$$[lf_k(S,R)] ::= \{a \in \mathcal{D}_{p_k(S)} | p_k(S)(a) \subset [R]\}.$$

Ha egy program leggyengébb előfeltételét minden A-beli részhalmazra megadjuk, akkor nyilván teljesen ismerjük a program viselkedését, hiszen ezzel a programfüggvényt is ismerjük. Valójában egy program viselkedését nem szükséges teljes egészében ismernünk, hanem csak bizonyos, céljainktól függő feltételek között.

Amikor programot készítünk, valamilyen célt akarunk elérni: azt, hogy a program valamilyen feltételeknek eleget tevő állapotban termináljon. Ha a program leggyengébb előfeltételét megadjuk a kedvező végállapotokat kijelölő feltételekre, akkor a számunkra fontos viselkedését ismerjük a programfüggvény megadása nélkül is.

10 A leggyengébb előfeltétel tulajdonságainak igazolása

Az a.) tulajdonság igazolása

Tegyük fel, hogy létezik $a \in A : a \in [lf(S, hamis)]$, azaz $p(S)(a) \subset [hamis] = \emptyset$. Ez pedig ellentmond annak, hogy $a \in \mathcal{D}_{p(S)}$ miatt a definíció szerint $p(S)(a) \neq \emptyset$. $\bowtie$

Az b.) tulajdonság igazolása

Legyen $a \in [lf(S,Q)]$, azaz $p(S)(a) \subset [Q]$. Mivel $[Q] \subset [R]$, ezért $p(S)(a) \subset [R]$, azaz $a \in [lf(S,R)]$. $\bowtie$

A c.) tulajdonság igazolása

$$\begin{aligned}
&[lf(S,Q)] \cap [lf(S,R)] = \\
&\{a \in A | p(S)(a) \subset [Q]\} \cap \{a \in A | p(S)(a) \subset [R]\} = \\
&\{a \in A | p(S)(a) \subset [Q] \wedge p(S)(a) \subset [R]\} = \\
&\{a \in A | p(S)(a) \subset [Q \wedge R]\}.
\end{aligned}$$

$\bowtie$

<u>*Az d.) tulajdonság igazolása*</u>

$$
\begin{aligned}
&[lf(S,Q)] \cup [lf(S,R)] = \\
&\{a \in A | p(S)(a) \subset [Q]\} \cup \{a \in A | p(S)(a) \subset [R]\} = \\
&\{a \in A | p(S)(a) \subset [Q] \vee p(S)(a) \subset [R]\} \subset \\
&\{a \in A | p(S)(a) \subset [Q \vee R]\}.
\end{aligned}
$$

$\bowtie$

Ha $p(S)$ függvény, akkor itt is egyenlőség van.

11 A leggyengébb előfeltétel és a megoldás kapcsolata

<u>*Az a.) állítás igazolása*</u>

i.) A $\Rightarrow$ eset igazolása.
Legyen S megoldása F-nek. Ez a következő két összefüggés fennállását jelenti:
- $\mathcal{D}_F \subset \mathcal{D}_{p(S)}$,
- $\forall a \in \mathcal{D}_F : p(S)(a) \subset F(a)$.

Ekkor tetszőleges $a \in \mathcal{D}_F$-re fennáll, hogy $a \in \{a \in \mathcal{D}_F | p(S)(a) \subset F(a)\} = [lf(S, \mathcal{P}(F(a)))]$.

ii.) A $\Leftarrow$ eset igazolása.
Azt kell belátni, hogy az adott feltétel esetén S megoldása F-nek.
- Tetszőleges $a \in \mathcal{D}_F$-re fennáll, hogy $a \in [lf(S, \mathcal{P}(F(a)))] \subset \mathcal{D}_{p(S)}$, azaz $\mathcal{D}_F \subset \mathcal{D}_{p(S)}$.
- Tetszőleges $a \in \mathcal{D}_F$ esetén a feltétel szerint $a \in [lf(S, \mathcal{P}(F(a)))] = \{a \in \mathcal{D}_F | p(S)(a) \subset F(a)\}$, $p(S)(a) \subset F(a)$. $\bowtie$

<u>*A b.) állítás igazolása*</u>

Az a.) pont állítása alapján $\forall a \in \mathcal{D}_F : a \in [lf(S, \mathcal{P}(F(a)))] \Leftrightarrow \forall H \subset \mathcal{D}_{F^{-1}} : F^{-1}(H) \subset [lf(S, \mathcal{P}(H))]$.

i.) A $\Rightarrow$ eset igazolása.
Legyen $H \in \mathcal{D}_{F^{-1}}$ tetszőleges. Tekintsük az $F^{-1}(H) = \{a \in A | a \in \mathcal{D}_F \wedge F(a) = H\}$ halmazt. Bármely $a \in F^{-1}(H)$-ra a feltétel szerint $a \in [lf(S, \mathcal{P}(F(a)))] = [lf(S, \mathcal{P}(H))]$. Következésképpen $F^{-1} \subset [lf(S, \mathcal{P}(H))]$.

ii.) A $\Leftarrow$ eset igazolása.
Legyen $a \in \mathcal{D}_F$ tetszőleges. Tekintsük azt a $H \in \mathcal{D}_{F^{-1}}$-et, amelyre $H = F(a)$. Ekkor nyilván $a \in F^{-1}(H)$. A feltétel szerint $F^{-1}(H) \subset [lf(S, \mathcal{P}(H))] = [lf(S, \mathcal{P}(F(a)))]$, $a \in [lf(S, \mathcal{P}(F(a)))]$. $\bowtie$

12 A specifikáció tétele

Általában sok olyan program van, ami megoldása egy adott feladatnak. A programozás azt jelenti, hogy ezek közül legalább egyet előállítunk. Tehát a programozó nem az összes megoldó programra kíváncsi. Ezért számára nem fontos, hogy egy kritérium szükséges is legyen ahhoz, hogy a program megoldás legyen. A specifikáció tétele elégséges feltételt ad arra, hogy egy program megoldása legyen egy feladatnak.

A specifikáció tételének igazolása

Minden $a \in \mathcal{D}_F$-hez létezik olyan $b \in B$, hogy $a \in [Q_b]$, és mivel $[Q_b] \subset [lf(S, R_b)]$, $[Q_b] \subset \mathcal{D}_{p(S)}$, ezért $a \in \mathcal{D}_{p(S)}$. Másrészt, ha $a \in \mathcal{D}_F$ és $a \in [Q_b]$, akkor $p(S)(a) \subset [R_b]$ és $[R_b] \subset F(a)$, azaz $p(S)(a) \subset F(a)$. $\bowtie$

Egy program feladatát a specifikáció tétele alapján $Q_b \Rightarrow lf(S, R_b)$ $(\forall b \in B)$ formulával adjuk meg. A következő példában megmutatjuk, hogyan írjuk ezt fel változók segítségével.

Példa : Legyen a feladat az, hogy határozzuk meg két szám maximumát.

$$A = \mathcal{Z} \times \mathcal{Z} \times \mathcal{Z}, \quad x, y, m : \mathcal{Z}.$$
$$B = \mathcal{Z} \times \mathcal{Z}, \quad x', y' : \mathcal{Z}.$$
$$Q_{x',y'} = (x = x' \wedge y = y'),$$
$$R_{x',y'} = (m \geq x' \wedge m \geq y' \wedge (m = x' \vee m = y')).$$

Ez a specifikáció azt jelenti, hogy olyan S programot kell keresni, amire $\forall b \in B$-re

$$x(a) = x'(b) \;\wedge\; y(a) = y'(b) \Rightarrow$$
$$lf(S, m(a) \geq x'(b) \wedge m(a) \geq y'(b) \wedge (m(a) = x'(b) \vee m(a) = y'(b))).$$

13 A specifikáció tétele nem megfordítható

A specifikáció tétele nem megfordítható, azaz van olyan F feladat, S program, B, F_1 és F_2, hogy S megoldja a feladatot, de $\forall b \in B : Q_b \Rightarrow lf(S, R_b)$ nem teljesül. A relációk alábbi megválasztásával erre mutatunk példát: $A = \{1,2,3\}, B = \{a,b\}, F(1) = \{1,2\}, F(2) = \{1,2,3\}, F(3) = \{2,3\}, [Q_a] = \{1,2\}, [R_a] = \{1,2\}, [Q_b] = \{2,3\}, [R_b] = \{2,3\}, p(S)(1) = \{1\}, p(S)(2) = \{1,2,3\}, p(S)(3) = \{2,3\}$. És így nem teljesül, hogy $[Q_a] \subset [lf(S, R_a)]$.

14 Értékadás

Az értékadást általában változókkal írjuk fel. Ha $x_1, \ldots, x_n$ az állapottér változói, az $a := F(a)$ programot

$$x_1, \ldots, x_n := F_1(x_1, \ldots, x_n), \ldots, F_n(x_1, \ldots, x_n)$$

jelöli, kifejezve, hogy melyik változó "hogyan változik". Ha valamely i-re $F_i(x_1, \ldots, x_n) = x_i$ (és ez az esetek döntő többségében így van), akkor a jelölésből az x_i-t és $F_i(X_1, \ldots, x_n)$-t elhagyjuk. Ha így $x_k := F_k(x_1, \ldots, x_n)$-hez jutunk, akkor az értékadást egyszerűnek, egyébként szimultánnak nevezzük. Az is igen gyakori, hogy az F_j nem függ valójában minden x_i-től, ekkor tovább egyszerűsítjük a jelölést, és elhagyjuk a fölösleges x_i-ket.

Az egyszerű értékadásnak nagyon szemléletes jelentése van: felfoghatjuk, mint az állapottér egy tengelyén való mozgást.

Mivel az értékadás olyan program, aminek a programfüggvényét ismerjük, ezért ismerjük a leggyengébb előfeltételét is, s gyakran használjuk az alábbi jelölést:

$$lf(x_k := F_k(x_1, \ldots, x_n), R) = R^{x_k \leftarrow F_k(x_1, \ldots, x_n)}.$$

Ha pedig az értékadás parciális, akkor a következőt:

$$lf(x_k := F_k(x_1, \ldots, x_n), R) = F_k(x_1, \ldots, x_n) \neq \emptyset \ \wedge \ R^{x_k \leftarrow F_k(x_1, \ldots, x_n)}.$$

15 Programok szekvenciája

A szekvencia olyan konstrukció, hogy ha mindkét komponense csupa véges sorozatból áll, akkor a szekvenciájuk is ilyen. Az is igaz, hogy determinisztikus programok szekvenciája is determinisztikus.

A szekvencia jelölése sruktogrammal:

S
$S1$
$S2$

16 Elágazás

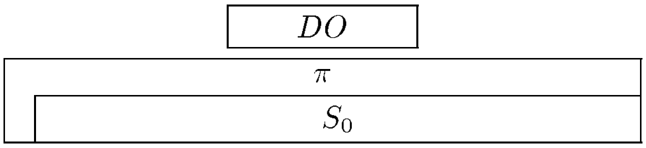
Az elágazás csupa determinisztikus programkomponens esetén is lehet nemdeterminisztikus, ha a feltételek igazsághalmazai nem diszjunktak. Az is lehet, hogy csupa véges sorozatból álló komponens esetén is tartalmaz végtelen (csupa azonos elemből álló) sorozatot is, hiszen a feltételek igazsághalmazainak úniója nem biztos, hogy kiadja az egész teret.

IF			
π_1	π_2	$\ldots$	π_n
S_1	S_2	$\ldots$	S_n

IF	
π	
S_1	S_2

17 Ciklus

A determinisztikus komponensből álló ciklus mindig determinisztikus, de véges komponens esetén is lehet a ciklus végtelen.



18 A szekvencia, az elágazás és a ciklus program

Bizonyitás

A definíció mindhárom esetben kimondja, hogy mindegyik reláció, azaz része $A \times A$-nak, és azt is, hogy a program b.) tulajdonsága ($\mathcal{D}_S = A$) teljesül. Ellenőrizzük a többi feltétel teljesülését!

a.) A szekvencia esetében, ha $\alpha \in S(a)(a \in A)$, akkor vagy $\alpha \in S_1(a)$, s ekkor nyilván teljesülnek a tulajdonságok, vagy $\alpha = \chi(\alpha_1, \alpha_2)$, és ekkor a χ definíciója miatt teljesül *c.*), de $\alpha_1 \in S_1$ miatt a.) is.

b.) Az elágazás esetében, mivel $\cup_{i=1}^{n} w_i(a)$ és $w(a)$ (ahol $a \in A$) nem lehet egyszerre üres, ezért vagy $\exists i \in [1, n] : \alpha \in S_i(a)$, vagy $\alpha = (a, a, a, \ldots)$. Nyilván mindkét esetben mindegyik tulajdonság teljesül.

c.) A ciklus esetében, ha $a \notin [\pi]$, akkor triviális a teljesülés. Ha $a \in [\pi]$, akkor $S(a)$ nem lehet üres, mivel S program. Tehát létezik A^{**}-beli sorozatoknak legalább egy olyan véges vagy végtelen $\alpha = (\alpha_1, \ldots)$ sorozata, ahol $\alpha_1 \in S(a)$, és $\alpha_{i+1} \in S(\tau(\alpha_i))$.
Másrészt minden ilyen tulajdonságú $\alpha = (\alpha_1, \ldots), \alpha_i \in A$ sorozatra: vagy $\exists i \in \mathcal{N} : \tau(\alpha_i) \in [\neg\pi]$, vagy nem. Ha egy sorozatra létezik ilyen i, akkor k legyen ezek közül a maximális. Ekkor $\chi_k(\alpha_1, \ldots, \alpha_k) \in DO(a)$, és rendelkezik a *c.*) tulajdonsággal. Ha egy α sorozatra nem létezik ilyen i, akkor, ha α véges, azaz $\alpha = (\alpha_1, \ldots, \alpha_m)$, a $\chi_m(\alpha) \in DO(a)$.
Ha α végtelen, a $\chi_\infty(\alpha) \in DO(a)$, és mindkét esetben a *c.*) pontnak megfelelő sorozatot kapunk.
Tehát $DO(a)$ nem üres, elemei rendelkeznek a *c.*) tulajdonsággal, és mivel $\alpha_1 \in S(a)$, az a.)-val is. $\bowtie$

19 A programkonstrukciók programfüggvényei

A szekvenciára vonatkozó állitás bizonyitása

Legyen $(a, a') \in A \times A$. $(a, a') \in p(S_2) \circ p(S_1)$ akkor és csak akkor, ha $\exists b \in A : (a, b) \in p(S_1) \wedge (b, a') \in p(S_2)$, azaz $S_1(a) \subset A^*$, és létezik $\alpha \in S_1(a)$, hogy $S_2(\tau(\alpha)) \subset A^*$ és létezik $\alpha' \in S_2(\tau(\alpha))$, amire $\tau(\alpha') = a'$, vagyis $(a, a') \in p(S)$. $\bowtie$

Az elágazásra vonatkozó állitás bizonyitása

Az elágazás definíciója szerint, egy $a \in A$ pontban $IF(a) \subset A^*$ pontosan akkor, ha a-ra legalább egy π_i igaz, és $\cup_{i=1}^{n} w_i(a) \subset A^*$. Ez nyilván teljesül, ha

$\forall i \in [1,n] : a \in [\pi_j] \Rightarrow a \in \mathcal{D}_{p(S)}$. Ekkor $p(IF)(a) = \tau(\cup_{i=1}^{n} w_i(a)) = \cup_{i=1}^{n} pw_i(a)$. $\bowtie$

A ciklusra vonatkozó állítás bizonyitása

Legyen $a \in A$. $a \in \mathcal{D}_{p(DO)}$ akkor és csak akkor, ha $DO(a) \subset A^*$, azaz, ha $DO(a) = \{(a) \in A^* | a \notin [\pi]\} \cup \{\alpha \in A^* | a \in [\pi] \wedge \exists n \in \mathcal{N} : \alpha = \chi_n(\alpha_1, \ldots, \alpha_n) \wedge \alpha_1 \in S_0(a) \wedge (\forall i \in [1,n] : \alpha_{i+1} \in S_0(\tau(\alpha_i))) \wedge \tau(\alpha_i) \in [\pi] \wedge \tau(\alpha_n) \notin [\pi]\}$. És ez éppen a π-re vonatkozó lezárt definíciója. $\bowtie$

20 A szekvencia levezetési szabálya

Az állitás bizonyitása

Ha $q \in [Q]$, akkor 2. miatt $p(S_1)(q) \in Q'$. De mivel 1. miatt minden $q' \in Q'$-re $p(S_2)(q' \subset [R]$, ezért $p(S_2) \circ p(S_1)(q) \subset [R]$, azaz $q \in [lf(S,R)]$. $\bowtie$

21 Az elágazás levezetési szabálya

Az állitás bizonyitása

Ha $q \in Q$, és létezik i, hogy $q \in [\pi_i]$, akkor mivel minden i-re $q \in [\pi_i]$-ből következik, hogy $q \in \mathcal{D}_{p(S_i)}$, és $p(S_i)(q) \subset [R]$, ezért $q \in \mathcal{D}_{p(IF)}$ és $p(IF)(q) \subset [R]$ is teljesül. $\bowtie$

22 A ciklus levezetési szabálya

Az állitás bizonyitása

Legyen $q \in [Q]$. Mivel $[Q] \subset [P]$, ezért $q \in [P] \cap [\neg\pi]$, vagy $q \in [P] \cap [\pi]$.

a) Legyen $q \in [P] \cap [\neg\pi]$. Ekkor a ciklus definíciója miatt $p(DO)(q) = (q)$, ész a 2. miatt $q \in [R]$, azaz $q \in [lf(DO,R)]$.

b) Jelöljük g-vel $p(S_0)$ leszűkítését $[\pi]$-re. Azaz $g = p(S_0) \cap ([\pi] \times A)$. Legyen $q \in [P] \cap [\pi]$. Ekkor $q \in \mathcal{D}_{p(S_0)}$ a 4. pont miatt. Ha $b \in g(q)$, akkor 4. miatt vagy $b \in [P] \cap [\pi]$, $b \in [P] \cap [\neg\pi]$, azaz $g(q) \subset ([P] \cap [\pi]) \cup ([P] \cap [\neg\pi])$. Ha feltesszük, hogy valamely k-ra $g^{(k)}(q) \subset ([P] \cap [\pi]) \cup ([P] \cap [\neg\pi])$, akkor ha $b \in g^{(k)}(q)$, vagy $b \in [\neg\pi] \cap [P]$, és 2. miatt $b \in [R]$, vagy $b \in [P] \cap [\pi]$, és akkor 4. miatt $p(S_0)(b) \subset ([P] \cap [\pi]) \cup ([P] \cap [\neg\pi])$, azaz $g^{(k+1)}(q) \subset ([P] \cap [\pi]) \cup ([P] \cap [\neg\pi])$. Így $g^{(k)}(q) \subset ([P] \cap [\pi]) \cup ([P] \cap [\neg\pi])$ minden k-ra igaz.
Másrészt, ha $q \in [P] \cap [\pi]$, akkor létezik $n \leq t(q)$, hogy $g(q)^{(n)} \in [\neg\pi]$. Tegyük fel, hogy nem igaz az állítás. Mivel 5. miatt, ha $b \in p(S_0)(q)$, akkor $t(b) < t(q)$ (hiszen $t(q) \leq t(q) - 1 + 1$ triviális fennállása alapján 5.-ből a $t(b) \leq t(q) - 1 < t(q)$ állítást kapjuk), ezért (mivel a t függvénynek $g^{(i)}(q)$-n $t(q)$ felső korlátja) $max_{b \in g^{(k+1)}(q)} t(b) < max_{b \in g^{(k)}(q)} t(b)$, azaz $max_{b \in g^{(t(q)+1)}(q)} t(b) < 0$, ez pedig ellentmond a 3. feltételnek, tehát $q \in \mathcal{D}_{\overline{p(S_0)|_\pi}}$, azaz $\mathcal{D}_{p(DO)}$ és $p(DO)(q) \in [R]$ a 2. tulajdonság miatt. $\bowtie$

Miért nevezzük ezeket a tételeket levezetési szabályoknak? Az alapvető kérdés minden programmal kapcsolatban az, hogy vajon megoldja-e az adott feladatot! Ez a kérdés a programkonstrukciókkal kapcsolatban is. A levezetési szabályokat megfogalmazó tételek lényege az, hogy ha a feltételeiknek megfelelő módon konstruálunk programot, akkor garantált, hogy azok megoldják a feladatot. Ha a tételek feltételeinek eleget tevő specifikációjú meghatározott feladatokat megoldó komponensekből indulunk ki, akkor az összetett program megoldja az eredeti feladatot.

Vegyük figyelembe, hogy a megoldandó feladatot a specifikáció tételének megfelelően a paramétertér segítségével adjuk meg Q_b és R_b elő- és utófeltételek formájában.

Így a szekvencia levezetési szabálya alapján azt mondjuk: ha S_1 és S_2 olyan programok, hogy a paramétertér minden elemére $Q_b \Rightarrow lf(S_1, Q'_b)$, és $Q'_b \Rightarrow lf(S_2, R_b)$, akkor $(S_1; S_2$ megoldja a Q_b, R_b párokkal megadott feladatot.

Hasonlóan az elágazás esetén: ha minden b paraméterre, mindegyik S_i programra $Q_b \wedge \pi_i \Rightarrow lf(S_i, R_b)$, akkor az $IF = (\pi_1 : S_1, \ldots, \pi_n : S_n)$ program megoldja a feladatot, ha minden b-re $\exists i : \pi_i$.

Így újrafogalmazhatjuk — valamivel egyszerűbben — a ciklusra vonatkozó tételt is. Azaz: ha minden $b \in B$-re teljesülnek az

1. $Q_b \Rightarrow P_b$,
2. $P_b \wedge \neg\pi \Rightarrow R_b$,
3. $P_b \wedge \pi \Rightarrow t_b > 0$

feltételek, és S_0 egy olyan program, amely megoldja a következő feladatot:

> Az állapottér az eredeti feladat állapottere, a változók szintén. A paramétertér legyen az eredeti bővítése, $B' = B \times \mathcal{Z}$, és az új paraméterváltozót jelöljük t'-vel. Az elő- és utófeltételpárok: $P_b \wedge \pi \wedge T' = t$, ill. $P_b \wedge T' > t$.

akkor a ciklus megoldja az eredeti, Q_b, R_b feltételpárokkal megadott feladatot.

23 Direktszorzat és rekord típusok

A direktszorzat típusok legtöbbször rekordok is, de nem mindig. Példa egy nem rekord direktszorzatra a racionális számok. A racionális számokat definiálhatjuk egészek direktszorzataként úgy, hogy ha $((a, b), t), ((a', b'), t') \in R$, akkor $a * b' = a' * b$.

24 Lineáris keresés

A feladat az, hogy az egész számok egy zárt intervallumán keressük meg a legkisebb adott tulajdonságú egész számot.

Ez a feladat valójában nagyon sok konkrét feladatot jelenthet. Például ilyenek "az adott intervallumba eső legkisebb prímszám megkeresése", "két azonos értelmezési tartományú függvény típusú változó első eltérő elemének kijelölése" (azaz a közös értelmezési tartományból ki kell választani a legkisebb olyan elemet, aminek az a tulajdonsága, hogy a hozzá tartozó függvényértékek különbözők), vagy "egy függvény típusú változónak melyik az első olyan helye, ahol a függvényérték az értékrészlet egy rögzített elemével egyenlő". A példákat még sokáig lehetne sorolni. Az a közös bennük, amit a feladatban megfogalmaztunk.

Először egy nagyon egyszerű esetet vizsgálunk. Legyen β egy feltétel az $A = \mathcal{Z}$ állapottéren, és $[m,n] \subset \mathcal{Z}$ olyan zárt intervallum, amire $[\beta] \cap [m,n] \neq \emptyset$. Keressük meg a legkisebb $i \in [m,n] \cap [\beta]$ elemet !

Állapottérnek az $A = \mathcal{Z} \times \mathcal{Z} \times \mathcal{Z}$ -t választjuk, és a megfelelő változók $m, n, i : \mathcal{Z}$. A paramétertér legyen $B = \mathcal{Z} \times \mathcal{Z}$, és a változói $m', n' : \mathcal{Z}$. A program specifikációja ezután a következő elő- és utófeltételekkel adódik:

$$Q_{m',n'} = (m = m' \wedge n = n' \wedge [m,n] \cap [\beta] \neq \emptyset),$$
$$R_{m',n'} = (Q_{m',n'} \wedge i \in [m,n] \wedge \beta(i) \wedge \forall j \in [m, i-1] : \neg\beta(j))$$

Az R állítást lényegében úgy foghatjuk föl, hogy az az $[m,i]$ intervallum minden eleméről állít valamit. Az ilyen esetekben gyakran ciklus oldja meg a feladatot. Ezért most is ezzel próbálkozunk.

Ha ciklusra akarjuk levezetni a programot, akkor az első dolgunk egy invariáns tulajdonságot keresni. Az invariáns tulajdonságot mindig úgy próbáljuk meghatározni, hogy az utófeltételt mint speciális esetet kaphassuk meg belőle. Azt is mondjuk, hogy az invariáns tulajdonság az utófeltétel általánosítása, vagy más szóval gyöngítése.

$R_{m',n'}$ kézenfekvő gyöngítése, ha elhagyjuk a $\beta(i)$ tényezőt. Válasszuk így az invariáns tulajdonságot:

$$P_{m',n'} = (Q_{m',n'} \wedge i \in [m,n] \wedge \forall j (m \leq j < i) : \neg\beta(j)).$$

Ezután a ciklusra vonatkozó levezetési szabályban szereplő feltételek teljesülését biztosítjuk.

Az első, $Q \Rightarrow P$ nem teljesül, ezért egy szekvenciává bontjuk először a programot.

$$S ::= (S_1; S_2),$$

ahol S_1 specifikációja az előzőeknek megfelelően $Q_{m',n'} \Rightarrow lf(S_1, P_{m',n'})$ lesz. Most azt kell meggondolnunk, lehetséges-e P -t úgy specializálni, szűkíteni, hogy az (könnyen) teljesíthető legyen (S_1 -el). A $Q_{m',n'} \wedge i = m$ állításból következik $P_{m',n'}$. Ezért S_1 specifikációja $Q_{m',n'} \Rightarrow lf(S, Q_{m',n'} \wedge i = m)$ lehet. Mivel pedig tetszőleges R -re $lf(i := m, R) = R^{i \leftarrow m}$, így $lf(i := m, Q_{m',n'} \wedge i = m) = (Q_{m',n'} \wedge m = m) = Q_{m',n'}$, azaz azt kaptuk, hogy $S_1 ::= i := m$.

Ezután vizsgáljuk a második feltétel teljesülését: $P \wedge \neg\pi \Rightarrow R$. Mivel $P_{m',n'} \wedge \beta(i) = R_{m',n'}$, ezért π -nek $\neg\beta(i)$ -t választva a második feltétel is teljesül, hiszen β -ról feltettük, hogy megengedett feltétel, a negáció pedig megengedett művelete a feltételek képzésének.

A harmadik feltétel teljesülését elérhetjük, ha t -nek $(n-i)$ -t választjuk, hiszen egyrészt $P_{m',n'}$ -ből következik, hogy $i \in [m,n]$, tehát $i \leq n$, másrészt $\neg\beta(i)$ miatt az egyenlőség sem teljesül.

Most már csak az van hátra, hogy olyan ciklusmagot válasszunk, amire teljesül a negyedik és az ötödik feltétel. A t függvény csökkentését elérhetjük az $i := i+1$ programmal, mert $P_{m',n'} \wedge \pi \wedge t \leq t_0+1 \Rightarrow lf(i := i+1, t \leq t_0)$ minden t_0 értékre, mivel

$$lf(i := i+1, t \leq t_0) = (n-i \leq t_0)^{i \leftarrow i+1} = (n-i-1 \leq t_0) = (n-i \leq t_0+1).$$

Másrészt

$$lf(i := i+1, P_{m',n'}) = P_{m',n'}^{i \leftarrow i+1} = Q_{m',n'} \wedge i+1 \in [m,n] \wedge \forall j \in [m,i] : \neg\beta(j) =$$
$$Q_{m',n'} \wedge i+1 \in [m,n] \wedge \beta(i) \wedge \forall j \in [m,i-1] : \neg\beta(j).$$

Könnyen látható, hogy az első és a harmadik tényező közvetlenül adódik $P_{m',n'}$ -ből, a negyedik π -ből, a második $P_{m',n'}$ -ből ($i \in [m,n]$ és $\forall j \in [m,i]$: $\neg\beta(j)$) és abból, hogy $Q_{m',n'}$ miatt $i \neq n$. Tehát ($GLO\ m\ :\ \mathcal{Z},\ \ VIR\ i :\ \mathcal{Z}$) az

<table>
<tr><td colspan="2">$i := m$</td></tr>
<tr><td colspan="2">$\neg\beta(i)$</td></tr>
<tr><td></td><td>$i := i+1$</td></tr>
</table>

programra és tetszőleges m', n' értékekre $Q_{m',n'} \Rightarrow lf(S, R_{m',n'})$, azaz S megoldás.

A kapott programra a *lineáris keresés első változata* néven fogunk hivatkozni.

Azt is mondhatnánk, hogy most beláttuk a következő tételt :

Az S program megoldása a keresési feladatnak.

Egy adott programról nagyon nehéz lenne eldönteni, hogy az megoldása-e valamilyen feladatnak. Ezért az állítást nem ilyen formában mondtuk ki, hanem bizonyítás, levezetés közben állítottuk elő a tételt.

Most egy kicsit általánosítjuk a feladatot, és nem kötjük meg, hogy β megengedett feltétel legyen. A programot az előző levezetéshez hasonlóan készítjük el, csak egy kicsit másképpen választjuk meg az invariáns tulajdonságot.

Legyen most az invariáns tulajdonság

$$P'_{m',n'} = (Q_{m',n'} \wedge m-1 \leq i \leq n \wedge (\forall j \in [m,i-1] : \neg\beta(j)) \wedge$$
$$\neg l = \forall j \in [m,i] : \neg\beta(j)).$$

A változás az eredeti invariáns tulajdonsághoz képest annyi, hogy a második állításban nem követeljük meg, hogy $i \geq m$ is teljesüljön, és kiegészítettük egy negyedik tényezővel. Most az $S_1 ::= i, l := m-1, \downarrow$ választással biztosíthatjuk $P'_{m',n'}$ teljesülését.

Mivel $P'_{m',n'} \wedge l \Rightarrow R_{m',n'}$, ezért ciklusfeltételnek $\neg l$ választható. A termináló függvény ugyanaz, mint az előbb. A ciklusmagot gyakran állítjuk elő mint szekvenciát. A szekvencia egyik tagjától azt várjuk, hogy csökkentse a termináló függvény értékét, a másiktól pedig azt, hogy biztosítsa az invariáns tulajdonság megmaradását. Mivel sokszor fordul elő, hogy a termináló függvény csökkentése nagyon egyszerű programmal elvégezhető, ezért az itt bemutatott eljárás nagyon gyakran használható.

Legyen $S_3 ::= (S_3'; S_3'')$. Mint az előzőekben már láttuk, az $i := i+1$ program csökkenti t -t, ezért S_3'' -at így választjuk meg. Ekkor

$$
\begin{aligned}
&lf(i := i+1, P'_{m',n'}) = P'^{\,i\leftarrow i+1}_{m',n'} = \\
&\quad Q_{m',n'} \wedge i+1 \in [m-1,n] \wedge (\forall j \in [m,i-1] : \neg\beta(j)) \wedge \neg\beta(i) \wedge \\
&\qquad \neg l = ((\forall j \in [m,i] : \neg\beta(j)) \wedge \neg\beta(i+1)) = \\
&\quad Q_{m',n'} \wedge i+1 \in [m-1,n] \wedge \forall j \in [m,i] : \neg\beta(j) \wedge \neg l = \neg\beta(i+1).
\end{aligned}
$$

Jelöljük a jobboldalt R' -vel. Így az S_3' programra a következő kikötést kaptuk :

$$P'_{m',n'} \wedge \neg l \wedge t = t_0 \Rightarrow lf(S_3', R' \wedge t = t_0).$$

Legyen $S_3' ::= l := \beta(i+1)$. Ekkor

$$
\begin{aligned}
lf(S_3', R' \wedge t = t_0) = Q_{m',n'} \wedge i+1 \leq n \wedge (\forall j \in [m,i] : \neg\beta(j)) \wedge \\
\neg\beta(i+1) = \neg\beta(i+1) \wedge n - i = t_0,
\end{aligned}
$$

ez pedig következik $P_{m',n'} \wedge \neg l \wedge t = t_0$ -ból, mert $Q_{m',n'}$ tényezője $P_{m',n'}$ -nek, $P_{m',n'} \wedge \neg l \wedge Q_{m',n'}$ miatt $i < n$, a harmadik tényező következik $P_{m',n'} \wedge \neg l$ -ből, a negyedik azonosan igaz, az ötödik azonos $t = t_0$ -al. Tehát ($GLO\ m : \mathcal{Z}$, $VIR\ i : \mathcal{Z}$, $LOC\ l : \mathcal{L}$) a programunk

$i, l := m-1, \downarrow$	
$\neg l$	
	$l := \beta(i+1)$
	$i := i+1$

Ezt az eredményt nevezzük a lineáris keresés második változatának.

További általánosítása a feladatnak, ha nem tesszük fel, hogy létezik a keresett tulajdonságú érték. Ilyenkor kezelhetjük a problémát úgy, hogy a keresett tulajdonságot kiegészítjük egy alkalmasan választott másik tulajdonsággal, amelyről tudjuk, hogy a szóbanforgó intervallumon valamikor teljesül.

Tegyük fel, hogy $\beta = \gamma \vee \delta$, és $[\delta] \cap [m,n] \neq \emptyset$. Keressük meg a legkisebb $i \in [m,n] \cap [\beta]$ elemet, és mondjuk meg azt is, hogy $\gamma(i)$ igaz-e ! Nagyon gyakori a $\delta(i) ::= (i = n)$ választás.

Az állapotteret $A = \mathcal{Z} \times \mathcal{Z} \times \mathcal{Z} \times \mathcal{L}$ -nek választjuk, a változók rendre m, n, i, u. A paramétertér — mint az előzőekben — $B = \mathcal{Z} \times \mathcal{Z}$, a paraméterváltozók m', n'.

$$
\begin{aligned}
&Q_{m',n'} = (m = m' \wedge n = n' \wedge [m,n] \cap [\delta] \neq \emptyset), \\
&R_{m',n'} = (Q_{m',n'} \wedge u = (\exists j \in [m,n] : \gamma(j) \wedge \forall k \in [m, j-1] : \neg\delta(k)) \wedge \\
&\qquad u \Rightarrow (i \in [m,n] \wedge \gamma(i) \wedge \forall j \in [m, i-1] : \neg\gamma(j)).
\end{aligned}
$$

Az előző levezetéshez hasonlóan ciklussal próbálkozunk. Bővítsük az állapotteret egy új $\mathcal{L}$ komponenssel és legyen $v : \mathcal{L}$ lokális változó.

$$\begin{aligned}P_{m',n'} = (Q_{m',n'} \wedge i \leq n \wedge \neg u = \forall j \in [m,i] : \neg\gamma(j) \wedge \\ \neg v = \forall j \in [m,i] : \neg\delta(j) \wedge \forall j \in [m,i-1] : \neg\beta(j))\end{aligned}$$

Az $i, u, v := m-1, \downarrow, \downarrow$ program garantálja hogy $P_{m',n'}$ teljesül. Erről helyettesítések után könnyen meggyőződhetünk.

A $P_{m',n'} \wedge \neg(\neg u \wedge \neg v) = P_{m',n'} \wedge (u \vee v)$ -ből következik $R_{m',n'}$. Ugyanis, ha u igaz, akkor $\gamma(i) \wedge i \in [m,n]$, az invariáns tulajdonság utolsó része miatt pedig minden i -nél kisebb j -re $\gamma(j)$ és $\delta(j)$ is hamis. Ha u hamis, akkor — π hamis volta miatt — v igaz, következésképpen $\delta(i)$ is az. Így valóban nincs olyan j, amire γ igaz és minden kisebb k -ra δ hamis. Tehát a ciklusfeltétel

$$\neg u \wedge \neg v.$$

Termináló függvénynek a $t = n - i$ most is megfelel, a csökkentésére ugyancsak jó az $i := i+1$ program. A $P_{m',n'} \wedge (\neg u \wedge \neg v) \wedge (t = t_0) \Rightarrow lf(S, P_{m',n'}^{i \leftarrow i+1} \wedge t = t_0)$ követelményt az

$$u, v := \gamma(i+1), \delta(i+1)$$

értékadás kielégíti, erről a helyettesítések elvégzése után könnyen meggyőződhetünk. Tehát a program a következő ($GLO\ m\ :\ \mathcal{Z},\ \ VIR\ i\ :\ \mathcal{Z},\ \ u\ :\ \mathcal{L},\ \ LOC\ v\ :\ \mathcal{L}$) :

$i, u, v := m-1, \downarrow, \downarrow$
$\neg u \wedge \neg v$
$u, v := \gamma(i+1), \delta(i+1)$
$i := i+1$

Ezt a programot nevezzük a lineáris keresés harmadik változatának. Jelentősége abban van, hogy elég általánosan oldja meg a feladott problémát. Alkalmas például akkor is a keresésre, ha nem biztos, hogy található az adott intervallumban a keresett tulajdonságú érték. Ha γ -nak a keresett tulajdonságot választjuk és $\delta(i) = (i = n)$, a harmadik változat megoldja a feladatot.

A program azonban ennél általánosabban is használható. Például olyan feladatok megoldására, mint hogy "keressünk egy adott prímszámnál kisebb olyan számot, aminek legalább három prímosztója van".

Még egy nagyon fontos észrevételt teszünk a feladattal kapcsolatban, ami azonban nem csak rá vonatkozik.

A β állításról az mondtuk, hogy az az egész számokra vonatkozik. Ezonban nem zárja ki, hogy β tetszőleges típusú változóktól függjön. Ha az $(x_1, ..., x_n, i)$ változóktól függ, a Q előfeltételt kiegészítjük az $x_1 = x_1' \wedge ... \wedge x_n = x_n'$ állításokkal, ahol x_i' -k a megfelelő paraméterváltozók. Mivel $R = (Q \wedge ...)$ volt, ezért ezek az állítások R -ben is szerepelnek. Ilyenkor β paraméterezett állításnak tekinthető.

$\boxed{25}$ **Logaritmikus keresés**

Legyen H egy olyan halmaz, amin értelmezve van egy rendezési reláció. Legyen $f : \mathcal{Z} \to H$ monoton növekedő függvény. A feladat az, hogy eldöntsük egy adott $h \in H$ értékről, hogy felveszi-e azt az f függvény egy adott $[m, n] \subset \mathcal{Z}$ intervallumon, és ha igen, akkor adjuk meg egy olyan pontját, ahol a függvényérték h.

Az állapottér legyen $A = \mathcal{Z} \times \mathcal{Z} \times \mathcal{Z} \times H \times \mathcal{L}$, a megfelelő változók m, n, i, h, l. A paramétertér $B = \mathcal{Z} \times \mathcal{Z} \times H$, a paraméterváltozók m', n', h'. Az elő- és utófeltételek

$$Q_{m',n',h'} = (m = m' \wedge n = n' \wedge h = h' \wedge \forall k, j \in \mathcal{Z} : k \leq j \Rightarrow f(k) \leq f(j)),$$
$$R_{m',n',h'} = (Q_{m',n',h'} \wedge l = (\forall j \in [m, n] : f(j) = h) \wedge l \Rightarrow f(i) = h).$$

A feladatot megoldhatnánk mint lineáris keresési feladatot, ha $\gamma(i) ::= (f(i) = h)$ és $\delta(i) ::= (i = n)$. Ez a megoldási út járható, de nem célszerű, mert nem veszi figyelembe a függvény monotonitását, pedig ennek a tulajdonságnak a kihasználása jobb programhoz vezet. A jobb itt most azt jelenti, hogy kevesebb lépésben találja meg a keresett elemet.

(Megjegyzés. Az, hogy két program közül melyik a jobb, nagyon bonyolult kérdés, és a jelenlegi ismeretek alapján nem is tudunk rá válaszolni. Bizonyos körülmények között lehet az a program jobb, ami más feltételek mellett rosszabb volt.)

A ciklus invariáns tulajdonsága a lineáris keresésnél az volt, hogy a keresett elem biztosan nincs az $[m, i]$ intervallumban. Hasonló meggondolással próbálkozunk most is:

$$\begin{aligned} P_{m',n',h'} = (Q_{m',n',h'} \wedge [u, v] \subset [m, n] \wedge \forall j \in [m, n] - [u, v] : f(j) \neq h \\ \wedge\, l = (i \in [u, v] \wedge f(i) = h)). \end{aligned}$$

Az $u, v, i, l := m, n, m - 1, \downarrow$ értékadás után $P_{m',n'}$ nyilvánvalóan igaz.

$P_{m',n'} \wedge l$ -ből következik $R_{m',n'}$, mert $f(i) = h \wedge i \in [m, n]$. Ha l hamis, akkor ahhoz, hogy $R_{m',n'}$ következzen $P_{m',n'}$ -ből még az szükséges, hogy az $[u, v]$ intervallum üres legyen, hiszen a $\forall j \in [m, n] - [u, v] : f(j) \neq h$ az $R_{m',n'}$ második állításának jobb oldaláról mondja ki, hogy hamis. Tehát a ciklusfeltétel: $\neg(l \vee u > v) = \neg l \wedge u \leq v$.

Termináló függvénynek a még vizsgálandó intervallum hosszát választjuk: $t = \begin{cases} u - v + 1, & \text{ha } \neg l \\ 0, & \text{ha } l \end{cases}$ A ciklusfeltételből következik, hogy $t > 0$.

A következő feladatunk a t, azaz az $[u, v]$ intervallum hosszának csökkentése. Ha most u -t növelnénk eggyel, a lineáris kereséshez jutnánk. Kihasználva az f függvény monotonitását azt mondhatjuk, hogy ha vesszük az $[u, v]$ intervallum bármely x pontját, akkor vagy $f(x) = h$, és akkor minden rendben van, vagy $f(x) < h$, és akkor csak $[x + 1, v]$ -ben lehet a keresett pont, vagy $f(x) > h$, és akkor csak $[u, x - 1]$ -ben lehet a keresett pont.

Ezért a ciklusmagot most egy olyan $S_1; S_2$ szekvenciának választjuk, ahol S_1 utófeltétele a következő Q' állítás:

$$Q' = (Q \wedge [u,v] \subset [m,n] \wedge \forall j \in [m,n] - [u,v] : f(j) \neq h \wedge \neg l \wedge u \leq v \wedge i \in [u,v]).$$

Ekkor $Q' \wedge f(i) = h \Rightarrow lf(l :=\uparrow, P_{m',n'}) = P^{l \leftarrow \uparrow}_{m',n'}$ nyilvánvalóan teljesül, $Q' \wedge f(i) < h \Rightarrow lf(u := i+1, P_{m',n'})$ és $Q' \wedge f(i) > h \Rightarrow lf(v := i-1, P_{m',n'})$ mivel i belső pontja volt $[u,v]$ -nek, szintén teljesül. Ha S_1 -nek $i := [(u+v)/2]$ -t választjuk, akkor $P_{m',n'} \wedge \pi \Rightarrow lf(S_1, Q')$, amiről helyettesítéssel meggyőződhetünk. Mivel S_1 nem változtatja sem u -t, sem v -t sem l -et, ezért a termináló függvényt sem. Az $l :=\uparrow, u := i+1$ és $v := i-1$ programok — mivel i belső pont — egyaránt csökkentik t -t. Tehát (GLO m, n : $\mathcal{Z}$, GLO h : H, VIR l : $\mathcal{L}$, i : $\mathcal{Z}$, LOC u, v : $\mathcal{Z}$) az

<table>
<tr><td colspan="4">$u, v, i, l := m, n, m-1, \downarrow$</td></tr>
<tr><td colspan="4">$\neg l \wedge (u \leq v)$</td></tr>
<tr><td rowspan="3"></td><td colspan="3">$i := [(u+v)/2]$</td></tr>
<tr><td>$f(i) < h$</td><td>$f(i) > h$</td><td>$f(i) = h$</td></tr>
<tr><td>$u := i+1$</td><td>$v := i-1$</td><td>$l :=\uparrow$</td></tr>
</table>

program megoldása a feladatnak.

Megjegyezzük, hogy az $i := [(u+v)/2]$ helyett bármilyen $i :\in [u,v]$ program is megfelelne. Ez a választás csökkenti a leggyorsabban az intervallum hosszát. Ha lineárisan keresünk, a vizsgálatok átlagos száma az elemek számával, logaritmikus keresés esetén az elemek számának logaritmusával arányos. Innen erednek az elnevezések.

26 Maximumkeresés

Legyen H egy tetszőleges rendezett halmaz és $f : \mathcal{Z} \rightarrow H$ egy függvény. Feladatunk az, hogy egy adott $[u,v] \subset \mathcal{Z}$ intervallumban keressük meg az f függvény maximumát és egy olyan helyét, ahol ezt a maximumértéket felveszi.

Az állapottér $A = H \times \mathcal{Z} \times \mathcal{Z} \times \mathcal{Z}$, a megfelelő változók $m : H, u, v, i : \mathcal{Z}$. A paramétertér $B = \mathcal{Z} \times \mathcal{Z}$ és a változói $u', v' : \mathcal{Z}$. Az elő- és utófeltételek:

$$Q_{u',v'} = (u = u' \wedge v = v'),$$
$$R_{u',v'} = (Q_{u',v'} \wedge i \in [u,v] \wedge m = f(i) \wedge \forall j \in [u,v] : f(j) \leq f(i)).$$

A második és harmadik tényező azt jelenti, hogy i maximumhely $[u,v]$ -ben, a negyedik pedig azt, hogy m a maximum értéke.

Az előző feladatokhoz hasonlóan gondolkodunk: nem követeljük meg, hogy m az egész $[u,v]$ intervallumon maximum legyen, hanem annak csak egy részén. Azaz ismét ciklussal próbálkozunk, ahol az invariáns tulajdonság:

$$P_{m',n'} = (Q_{u',v'} \wedge k \in [u,v] \wedge i \in [u,k] \wedge m = f(i) \wedge \forall j \in [u,k] : f(j) \leq f(i)).$$

Az invariáns tulajdonság teljesülése az $i, k, m := u, u, f(u)$ programmal elérhető. $P_{m',n'} \wedge k = v \Rightarrow R_{m',n'}$, ezért a ciklusfeltétel $k \neq v$. Ha termináló függvénynek $v - k$ -t választjuk, $P_{m',n'} \wedge k \neq v$ -ből következik, hogy $v - k$ pozitív; t csökkentésére a $k := k + 1$ program alkalmas.

Az

$$\begin{aligned} lf(k := k+1, P_{m',n'}) = (&Q_{u',v'} \wedge k+1 \in [u,v] \wedge m = f(i) \wedge f(k+1) \leq f(i) \wedge \\ &\forall j \in [u,k] : f(j) \leq f(i)) \end{aligned}$$

feltétel teljesülését kell még biztosítani. Jelöljük a jobboldalt Q' -vel.

Mivel $P_{m',n'} \wedge k \neq v$, ezért Q' első három tényező közvetlenül következik, csak a másik kettővel kell foglalkozni. Ha $f(k+1) \leq f(i)$, akkor Q' teljes egészében igaz, ha nem , akkor az $i, m := k+1, f(k+1)$ program biztosítja a teljesülését. Ezek a programok t értékét nem változtatják, ezért a megoldó program $(GLO\ u,\ v\ :\ \mathcal{Z},\ \ VIR\ i\ :\ \mathcal{Z},\ \ m\ :\ H,\ \ LOC\ k\ :\ \mathcal{Z})$

<table>
<tr><td colspan="3">$i, k, m := u, u, f(u)$</td></tr>
<tr><td colspan="3">$k \neq v$</td></tr>
<tr><td rowspan="3"></td><td colspan="2">\ $m < f(k+1)$ /</td></tr>
<tr><td>$i, m := k+1, f(k+1)$</td><td>$SKIP$</td></tr>
<tr><td colspan="2">$k := k+1$</td></tr>
</table>

27 Adott tulajdonsággal rendelkező elemek megszámolása

Legyen β egy az egész számokon értelmezett logikai függvény. A feladat az, hogy számoljuk meg, hány helyen igaz β az $[m,n] \subset \mathcal{Z}$ intervallumban.

Az állapottér legyen $A = \mathcal{Z} \times \mathcal{Z} \times \mathcal{N}_0$, a változók $m, n : \mathcal{Z}, s : \mathcal{N}_0$. A paramétertér $B = \mathcal{Z} \times \mathcal{Z}$, a változói $m', n' : \mathcal{Z}$. Az elő- és utófeltételek

$$\begin{aligned} Q_{m',n'} &= (m = m' \wedge n = n'), \\ R_{m',n'} &= (Q_{m',n'} \wedge s = \sum_{i=m}^{n} \chi(\beta(i))), \end{aligned}$$

ahol $\chi : \mathcal{L} \to \mathcal{N}_0, \chi(\uparrow) = 1, \chi(\downarrow) = 0$.

Kézenfekvő, hogy a megoldásra ciklussal próbálkozzunk. Az $R_{m',n'}$ -ben s az összes $[m,n]$ -beli igaz helyek számát jelenti. Ennél kevesebbet mondunk ki invariáns tulajdonságnak: csak $[m,n]$ egy részére legyen igaz az állítás.

$$P_{m',n'} = (Q_{m',n'} \wedge k \leq n \wedge s = \sum_{i=m}^{k} \chi(\beta(i))).$$

Nyilván, ha $k < m$, akkor $s = 0$, ezért a $k, s := m - 1, 0$ program biztosítja $P_{m',n'}$ teljesülését.

A ciklusfeltétel $k \neq n$, mivel $P_{m',n'} \wedge k = n \Rightarrow R_{m',n'}$. Termináló függvénynek $n - k$ megfelel, mert $k \leq n \wedge k \neq n$ miatt $n - k$ pozitív. Csökkentése a szokásos módon, a $k := k + 1$ programmal történik.

Most már csak a $P^{k \leftarrow k+1}_{m',n'}$ állítás teljesülését kell elérni.

$$P^{k \leftarrow k+1}_{m',n'} = (Q_{m',n'} \wedge k + 1 \leq n \wedge s = \chi(\beta(k+1)) + \sum_{i=m}^{k} \chi(\beta(i)))$$

$P_{m',n'} \wedge k \neq n$ -ből az első két tényező következik, és ha $\beta(k+1)$ hamis, a harmadik is. Ha $\beta(k+1)$ igaz, az $s := s+1$ program biztosítja a harmadik tényező teljesülését. Tehát a program ($GLO\ m,\ n : \mathcal{Z},\ \ VIR\ s : \mathcal{N}_0,\ \ LOC\ k : \mathcal{Z}$):

<table>
<tr><td colspan="3">$k, s := m - 1, 0$</td></tr>
<tr><td colspan="3">$k \neq n$</td></tr>
<tr><td rowspan="3"></td><td colspan="2">\ $\beta(k+1)$ /</td></tr>
<tr><td>$s := s + 1$</td><td>$SKIP$</td></tr>
<tr><td colspan="2">$k := k + 1$</td></tr>
</table>

28 Rendezés

Legyen T tetszőleges rendezett típus, és $X = seq(T)$, legyen továbbá $x : X$. Rendezzük át az x sorozatot úgy, hogy az elemei növekedő sorrendben legyenek.

Az állapottér $A = X$, a változó $x : X$. A paramétertér $B = X$, a változó x'. Az elő- és utófeltételek

$$Q_{x'} = (x = x'),$$
$$R_{x'} = (x \in perm(x') \wedge (\forall i, j \in [1, x.dom] : i \leq j \Rightarrow x_i \leq x_j))$$

Definiálunk két logikai függvényt a további állítások egyszerűbb leírásához.

$$\rho(x, i, j) = (\forall k, l(i < k < l \leq j) : x_k \leq x_l),$$
$$\mu(x, i, j) = (\forall k, l(1 \leq k \leq i < l \leq j) : x_k \leq x_l)$$

A ρ jelentése szavakban: az x_i-nél nagyobb indexű elemei j -ig rendezettek. A μ jelentése: az x -ben i -ig kisebb elemek vannak, mint i -től j -ig. ρ segítségével ismét felírjuk R -et:

$$R_{x'} = (x \in perm(x') \wedge \rho(x, 0, x.dom))$$

Ezután foglalkozunk a feladattal. A megoldás alapötlete az, hogy a sorozatot két szeletre bontjuk úgy, hogy az egyik szelet növekvően rendezett legyen, a másik szelet elemei pedig nem nagyobbak a rendezett szelet elemeinél.

Ezt fejezi ki a P invariáns tulajdonság:

$$P_{x'} = (x \in perm(x') \wedge 0 \leq k \leq x.dom \wedge \rho(x, k, x.dom) \wedge \mu(x, k, x.dom)).$$

Mivel $k = x.dom$ esetén $P_{x'}$ igaz, ezért $P_{x'}$ teljesülését a $k := x.dom$ program biztosítja. A ciklusfeltétel $k \neq 0$, mert $P_{x'} \wedge k = 0 \Rightarrow R_{x'}$. Mivel $P_{x'} \wedge k \neq 0$ esetén k pozitív, k -t választjuk termináló függvénynek.

A ciklusmagot egy szekvenciaként állítjuk elő. A rendezetlen részből egy új rendezett szeletet állítunk elő, és ezt csatlakoztatjuk a rendezett szelethez. Legyen

$$Q' = (\rho(x,m,k) \wedge \mu(x,m,k) \wedge (0 \leq m < k \leq x.dom) \wedge P_{x'}).$$

Ha $P_{x'} \wedge k \neq 0 \Rightarrow lf(\text{"}UJSZELET\text{"}, Q')$, akkor az $(GLO\ x\ :\ X,\ \ LOC\ l,\ \ m\ :\ \mathcal{N}_0)$

$k := x.dom$	
$k \neq 0$	
	$UJSZELET$
	$k := m$

program megoldás, mert $m < k$ miatt t csökken, és $P_{x'}^{k \leftarrow m}$ következik Q' -ből.

Az "UJSZELET" program is ciklus lesz. "Elengedjük" Q' -ből azt, hogy az új szelet vége k, azaz, hogy csatlakozik a rendezett részhez:

$$P' = (\rho(x,m,l) \wedge \mu(x,m,l) \wedge m < l \wedge 1 \leq l \leq k \wedge P_{x'})$$

Az $m, l := 0, 1$ programra $P_{x'} \wedge k \neq 0 \Rightarrow lf(m, l := 0, 1, P')$, ami a helyettesítés után nyilvánvaló. Mivel $P' \wedge l = k \Rightarrow Q'$, ezért a ciklusfeltétel $l \neq k$. Termináló függvénynek így $k - l$ választható, amit az $l := l + 1$ programmal csökkenthetünk. Ekkor

$$P'^{l \leftarrow l+1} = (\rho(x,m,l+1) \wedge \mu(x,m,l+1) \wedge m < l+1 \wedge 1 \leq l+1 \leq k \wedge P_{x'}).$$

Az utolsó állítás benne van P' -ben, az utolsó előtti a ciklusfeltétel miatt teljesül, és $m < l$ miatt $m < l+1$. Ezért csak az első két állítás teljesüléséről kell gondoskodni. Ha $x_{l+1} \geq x_l$, akkor nyilván $\rho(x,m,l+1)$ és $\mu(x,m,l+1)$ is igaz, ha $\rho(x,m,l)$ és $\mu(x,m,l)$ igaz volt. Ha $x_{l+1} < x_l$, az $m, x_l, x_{l+1} := l, x_{l+1}, x_l$ program után $\rho(x,m,l+1)$ és $\mu(x,m,l+1)$ igaz, $m < l+1$ is teljesül, és $P_{x'}$ is igaz marad, mert $l+1 \leq k$ és $x \in perm(x')$. Tehát az "UJSZELET" program a következő $(GLO\ x\ :\ X,\ \ k\ :\ \mathcal{N}_0,\ \ VIR\ m\ :\ \mathcal{N}_0,\ \ LOC\ l\ :\ \mathcal{N})$:

$UJSZELET$

$m, l := 0, 1$		
$l \neq k$		
	$x_{l+1} < x_l$	
	$m, x_l, x_{l+1} := l, x_{l+1}, x_l$	$SKIP$
	$l := l + 1$	

Ezzel kaptunk egy olyan programot, amelyik rendezi a sorozatot. Azonban más megoldó program is lehetséges. A logaritmikus keresést azért hoztuk létre, mert kihasználva a függvény monotonitását, a lineáris keresésnél bizonyos tekintetben jobb — kevesebb lépésben jut végállapotba — programot készítettünk.

A rendezés esetében is próbálkozhatunk jobb program készítésével.

A program elkészítésének alapgondolata most is nagyon egyszerű: a rendezett és rendezetlen részből álló sorozatban a rendezett rész hosszát növeljük, mégpedig úgy, hogy a rendezetlen rész első eleme kerüljön át a rendezett részbe.

Tehát az invariáns tulajdonság

$$P_{x'} = (\rho(x,0,l) \wedge 1 \leq l \leq x.dom \wedge x \in perm(x')).$$

Ez az állítás teljesül az $l := 1$ program után. Mivel $P_{x'} \wedge l = x.dom \Rightarrow R$, ezért a ciklusfeltétel $l \neq x.dom$. A termináló függvény $x.dom - l$, ami $P_{x'} \wedge l \neq x.dom$ miatt pozitív, és az $l := l+1$ program csökkenti. Most a $P_{x'} \wedge l \neq x.dom \Rightarrow lf(S_1, P_{x'}^{l \leftarrow l+1})$ programot kell elkészíteni, ahol $P_{x'}^{l \leftarrow l+1} = (\rho(x,0,l+1) \wedge 1 \leq l+1 \leq x.dom \wedge x \in perm(x'))$. A ciklusfeltétel miatt $l+1 \leq x.dom$. Az S_1 program legyen szintén ciklus P' invariáns tulajdonsággal:

$$P' = ((0 \leq k \leq l) \wedge \rho(x,0,k) \wedge \mu(x,k,l+1) \wedge x \in perm(x')).$$

Ha $k = l$, a P' következik $P_{x'}$ -ből. Ezért a $k := l$ program garantálja P' teljesülését.

Mivel $P' \wedge (k = 0 \vee x_{k+1} \geq x_k) \Rightarrow \rho(x,0,l+1) \wedge x \in perm(x')$, ezért a ciklusfeltétel $k \neq 0 \wedge x_{k+1} < x_k$. A termináló függvény k, csökkentése a $k := k-1$ programmal történik. A $p'^{k \leftarrow k+1}$ teljesülését az $x_k, x_{k+1} := x_{k+1}, x_k$ program garantálja. Tehát a teljes program ($GLO\ x : X,\ \ LOC\ l : \mathcal{N}_0$):

- $l := 1$
- $l \neq x.dom$
  - $S1$
  - $l := l+1$

$S1$

- $k := l$
- $k \neq 0 \wedge x_{k+1} < x_k$
  - $x_k, x_{k+1} := x_{k+1}, x_k$
  - $k := k-1$

Ez a program lényegében egyenértékű az előző változattal, azonban az $x_k, x_{k+1} := x_{k+1}, x_k$ program helyett kevesebb műveletet igénylő programot is választhatunk, ugyanis a legtöbb jelenlegi gép ún. szekvenciális gép, és a fenti szimultán értékadás csak három egyszerű értékadással valósítható meg.

A P' helyett válasszunk egy olyan P'' invariáns tulajdonságot, amiben nem követeljük meg az $x \in perm(x')$ teljesülését, csak azt, hogy $kon((x_1, ..., x_k), (e), (x_{k+2}, ..., x_{x.dom}) \in perm(x')$. Ekkor az $e, k := x_{l+1}, l$ program biztosítja P'' teljesülését, a ciklusfeltétel pedig $k \neq 0 \wedge e < x_k$, a ciklusmag pedig $x_{k+1} := x_k; k := k + 1$ lesz. Az $x_{k+1} := e$ után teljesül $P^{l \leftarrow l+1}$. Azaz $(GLO\ x : X,\ \ l : \mathcal{N},\ \ LOC\ k : \mathcal{N},\ \ e : T_0)$

$S1'$

$k, e := l, x_{k+1}$
$k \neq 0 \wedge e < x_k$
$x_{k+1} := x_k$ $k := k - 1$
$x_{k+1} := e$

Ez a program már olcsóbb műveleteket tartalmaz mint az előző, gyorsabban hajtódik végre a szekvenciális gépen.

Vajon ez már az ideális program? Korántsem. Ennek vizsgálata azonban már egy más tárgy feladata.

29 Elemenkénti feldolgozás — első változat

Először a legegyszerűbb esettel foglalkozunk, amikor a leképezés egyváltozós egyértékű függvény. Ezt a továbbiakban

$$z = f(x)$$

típusú feladatnak nevezzük.

Legyen tehát $f : X \to Y, X, Y \subset 2^H$ függvény. Az állapottér $A = A_1 \times A_2$, ahol $A1 = set(H_1), A_2 = set(H_2), H_1 \subset H, H_2 \subset H$. A változók legyenek $x : A_1, z : A_2$. A paramétertér $B = A_1$, változója $x' : A_1$, az elő- és utófeltételek

$$Q_{x'} = (x = x'),$$
$$R_{x'} = (z = f(x'))$$

Kihasználva f elemenkénti feldolgozhatóságát, programunk egy ciklus lesz a

$$P = (z \cup f(x) = f(x') \wedge z \cap f(x) = \emptyset)$$

invariáns tulajdonsággal.

Az $z := \emptyset$ program után P teljesül. Ciklusfeltételnek választható $x \neq \emptyset$ is, de $f(x) \neq \emptyset$ is, mivel $P \wedge f(x) = \emptyset \Rightarrow R$, és $x = \emptyset \Rightarrow f(x) = \emptyset$ az elemenkénti feldolgozhatóság miatt. A célszerűbb $f(x) \neq \emptyset$ mellett maradunk, ha az primitív feltétel. Termináló függvénynek x.dom megfelel, hiszen x nem lehet üres a ciklusfeltétel miatt.

A ciklusmag x egy elemének kiválasztásából és ennek feldolgozásából áll, azaz egy $S_1; S_2$ szekvencia lesz. A közbülső Q' állítás legyen

$$Q' = (P \wedge e \in x).$$

Ez az $S_1 ::= e : mem(x)$ programmal elérhető, hiszen

$$lf(S_1, Q') = (x \neq \emptyset \wedge s \in x \Rightarrow P^{e \leftarrow s} \wedge s \in x),$$

ez pedig következik $P \wedge f(x) \neq \emptyset$ -ból. Az S_2 program is egy szekvencia: $x := x \tilde{-} \{e\}$ csökkenti a t -t, ekkor

$$\begin{aligned} Q'' = {} & lf(x := x \tilde{-} \{e\}, P) = \\ & ((z \cup f(x \tilde{-} \{e\}) = f(x') \wedge z \cap f(x \tilde{-} \{e\}) = \emptyset \wedge x \cap \{e\} \neq \emptyset), \end{aligned}$$

és

$$\begin{aligned} & lf(z := z \tilde{\cup} f(\{e\}), Q'') = \\ & \quad z \cup f(\{e\}) \cup f(x \tilde{-} \{e\}) = f(x') \wedge z \cup f(\{e\}) \cap f(x \tilde{-} \{e\}) = \\ & \quad \emptyset \wedge x \cap \{e\} \neq \emptyset \wedge z \cap f(\{e\}) = \emptyset). \end{aligned}$$

Ez pedig Q' -ből következik, mert $e \in x$ és f elemenkénti feldolgozhatósága miatt $f(\{e\}) \cup f(x \tilde{-} \{e\}) = f(x)$ és $f(\{e\}) \cap f(x \tilde{-} \{e\}) = \emptyset$. Ezért csak azt kell belátni, hogy $z \cap f(\{e\}) = \emptyset$ és $z \cap f(x \tilde{-} \{e\}) = \emptyset$ is teljesül. Mivel azonban P -ben $z \cap f(x) = \emptyset$ és $f(\{e\})$ és $f(x \tilde{-} \{e\})$ is része $f(x)$ -nek, ezért metszetük z -nal üres. Tehát az $z = f(x)$ feladatot megoldó program ($GLO\ x : A_1,\ VIR\ z : A_2,\ LOC\ e_1 : H$):

$z := \emptyset$	
$f(x) \neq \emptyset$	
	$e : mem(x)$
	$z := z \tilde{\cup} f(\{e\})$
	$x := x \tilde{-} \{e\}$

Mit értünk el ezzel a programmal? Figyeljük meg, hogy az S programban már csak egyelemű halmazokra kell alkalmazni f -et, azaz f helyett elég egy olyan f' -vel foglalkoznunk, amelyik nem 2^{H_1} -ből képez 2^{H_2} -be, hanem csak H_1 -ből H_2 -be képez.

30 Elemenénti feldolgozás — második változat

Most az első változattal szemben azt tesszük fel, hogy az f kétváltozós függvény. Így az állapottér $A = A_1 \times A_2 \times A_3, A_i = set(H_i), H_i \subset H$ (i=1,2,3), a változók $x : A_1, y : A_2, z : A_3$. A paramétertér $B = A_2 \times A_3$, a változói $x' : A_1, y' : A_2$. Az elő- és utófeltételek

$$\begin{aligned} Q_{x',y'} &= (x = x' \wedge y = y'), \\ R_{x',y'} &= (z = f(x', y')). \end{aligned}$$

Ezt a változatot a $z = f(x, y)$ típusú feladatnak nevezzük.

A megoldó programot az előzőhöz hasonlóan kapjuk. Az invariáns tulajdonság kicsit kibővül, hogy a teljes diszjunktság teljesüljön:

$$P = (z \cup f(x, y) = f(x', y') \wedge f(x, y) \cap z = \emptyset \wedge (x' - x) \cap y = \emptyset \wedge (y' - y) \cap x = \emptyset).$$

A ciklusfeltétel $f(x, y) \neq \emptyset$, a termináló függvény $x \cup y$ elemeinek száma. A ciklusmag most is egy olyan szekvencia, aminek első tagja $e : mem(x \cup y)$, ami után a $Q' = P \wedge e \in x \cup y$ teljesül, hasonlóan mint az előző változatnál.

A szekvencia második tagja most egy elágazás lesz annak megfelelően, hogy $e \in x \cup y$ háromféleképpen fordulhat elő: e csak x -nek, csak y -nak, vagy mindkettőnek eleme. Könnyen látható, hogy igazak a következő állítások:

$$Q' \wedge e \in x \wedge e_s y \Rightarrow lf(z := z\tilde{\cup} f(\{e\}, \emptyset); x := x\tilde{-}\{e\}, P),$$
$$Q' \wedge e_s x \wedge e \in y \Rightarrow lf(z := z\tilde{\cup} f(\emptyset, \{e\}); y := y\tilde{-}\{e\}, P),$$
$$Q' \wedge e \in x \wedge e \in y \Rightarrow lf(z := z\tilde{\cup} f(\{e\}, \{e\}); x := x\tilde{-}\{e\}; y := y\tilde{-}\{e\}, P).$$

Pl. a harmadik teljesülése a helyettesítések elvégzése után:

$$\begin{aligned} Q' \wedge e \in x \wedge e \in y \Rightarrow\ & (z \cup f(\{e\}, \{e\}) \cup f(x\tilde{-}\{e\}, y\tilde{-}\{e\}) = \\ & f(x', y') \wedge f(x\tilde{-}\{e\}, y\tilde{-}\{e\}) \cap (z \cup f(\{e\}, \{e\})) = \\ & \emptyset \wedge (x' - (x - \{e\})) \cap (y - \{e\}) = \emptyset \wedge (y' - (y - \{e\})) \cap (x - \{e\}) = \\ & \emptyset \wedge x \cap \{e\} \neq \emptyset \wedge y \cap \{e\} \neq \emptyset \wedge z \cap f(\{e\}, \{e\}) = \emptyset). \end{aligned}$$

Figyelembe véve, hogy f tulajdonsága miatt $f(\{e\}, \{e\}) \cup f(x - \{e\}, y - \{e\}) = f(x, y)$ és $f(\{e\}, \{e\}) \cap f(x - \{e\}, y - \{e\}) = \emptyset$, továbbá $e, x\tilde{-}\{e\} \in x$ és $e, y\tilde{-}\{e\} \in y$ is igazak, ezért az előző implikáció valóban fennáll.

Tehát a második változat programja ($GLO\ x\ :\ A_1,\ \ y\ :\ A_2,\ \ VIR\ z\ :\ A_3,\ \ LOC\ e\ :\ H$):

$z := \emptyset$		
$f(x, y) \neq \emptyset$		
$e : mem(x \cup y)$		
$e \in x \wedge e \notin y$	$e \notin x \wedge e \in y$	$e \in x \wedge e \in y$
$z := z\tilde{\cup} f(\{e\}, \emptyset)$ $x := x\tilde{-}\{e\}$	$z := z\tilde{\cup} f(\emptyset, \{e\})$ $y := y\tilde{-}\{e\}$	$z := z\tilde{\cup} f(\{e\}, \{e\})$ $x := x\tilde{-}\{e\}$ $y := y\tilde{-}\{e\}$

31 Elemenkénti feldolgozás — harmadik változat

Most az első változatot úgy módosítjuk, hogy az f függvényről feltesszük, hogy (f_1, f_2) alakú. Azaz az állapottér $A = A_1 x A_2 x A_3, A_i = set(H_i \subset H)$

(i=1,2,3), a változók pedig $x : A_1, y : A_2, z : A_3$. A paramétertér $B = A_1$, a változója x'. Az elő- és utófeltételek

$$Q_{x'} = (x = x'),$$
$$R_{x'} = (y = f_1(x') \wedge z = f_2(x')).$$

Az első változatból kis átalakítással kapjuk, hogy

$$P = (y \cup f_1(x) = f_1(x') \wedge y \cap f_1(x) = \emptyset \wedge z \cup f_2(x) = f_2(x') \wedge z \cap f_2(x) = \emptyset).$$

Most a ciklusfeltétel $f_1(x) \neq \emptyset \vee f_2(x) \neq \emptyset$, vagy $x \neq \emptyset$. A termináló függvény ismét $x.dom$ lehet. Ezek után könnyen ellenőrizhető, hogy a megoldó program $(GLO\ x : A_1,\ VIR\ y : A_2,\ z : A_3,\ LOC\ e : H)$:

$y, z := \emptyset, \emptyset$	
$f_1(x) \neq \emptyset \vee f_2(x) \neq \emptyset$	
	$e : mem(x)$
	$y, z := y \tilde{\cup} f_1(\{e\}), z \tilde{\cup} f_2(\{e\})$
	$x := x \tilde{-} \{e\}$

32 Elemenkénti feldolgozás — általános változat

Különösebb probléma nélkül elkészíthető rögzített m -re, n -re a $z_i = f_i(x_1, ..., x_n)$ $(i = 1, ..., m)$ feladat programja is az előzőek mintájára. A levezetés formulái így alakulnak:

$$P = (\forall i \in [1, m] : (z_i \cup f_i(x_1, ..., x_n) = f_i(x'_1, ..., x'_n) \wedge$$
$$z_i \cap f_i(x_1, ..., x_n) = \emptyset) \wedge (\forall i, j \in [1, n] : (x'_i - x_i) \cap x_j = \emptyset)).$$

A ciklusfeltétel $f_1(x_1, ..., x_n) \cup ... \cup f_m(x_1, ..., x_n) = \emptyset$, vagy $\cup_{i=1}^{n} x_i \neq \emptyset$. A termináló függvény $\cup_{i=1}^{n} x_i$ elemeinek száma. A ciklusmagban levő elágazás "ágainak" száma $2^n - 1$. A program sémája a következő:

$y_1, ..., y_m := \emptyset, ..., \emptyset$			
$\cup_{i=1}^{m} f_i(x_1, ..., x_n) \neq \emptyset$			
	$e : mem(\cup_{i=1}^{n} x_i)$		
	$\backslash \cdots$	$\backslash$ $e \in x_{i_1} \wedge \ldots e \in x_{i_k} \wedge e \notin x_{i_{k+1}} \wedge e \notin x_{i_n}$	$\backslash \cdots$
	$\cdots$	$y := y \tilde{\cup} f(sl(\{i_1 \ldots i_k\}, e))$ $x_{i_1}, \ldots, x_{i_k} := x_{i_1} \tilde{-} \{e\}, \ldots, x_{i_k} \tilde{-} \{e\}$	$\cdots$

ahol $i_1, ..., i_n$ az $1, ..., n$ permutációja, $sl : 2^{\{1,...,n\}} \times H \to Y$,

$$sl(\{i_1, .., i_k\}, e)_i = \begin{cases} \{e\}, & \text{ha } i \in i_1, ..., i_k \\ \emptyset, & \text{ha } i \notin i_1, ..., i_k \end{cases}$$

az elágazás "ágainak" száma $2^n - 1$,

Megjegyezzük végül, hogy mindegyik változat levezethető elemenként feldolgozható reláció esetén is a függvényhez hasonló módon, értelemszerű módosításokkal. Ezt a feladatot az olvasóra bízzuk.