Bevezetés a programozásba[1]

Fóthi Ákos, Horváth Zoltán

2005. április 22.

ý

ELTE Informatikai Kar, 2005. A mű digitális megjelenítése az Oktatási Minisztérium támogatásával, a Felsőoktatási Tankönyv- és Szakkönyv-támogatási Pályázat keretében történt.

I. rész

Bevezetés a programozáshoz

1. fejezet

Alapfogalmak

Ebben a részben bevezetjük azokat a jelöléseket és alapvető definíciókat, amelyeket a továbbiakban gyakran fogunk használni. Ezek legnagyobb része középiskolából, vagy bevezető jellegű matematikai tanulmányokból ismert.

1.1. Halmazok

Először bevezetjük a matematikában gyakran használt halmazok jelöléseit.

$\mathbb{N}$	–	a természetes számok halmaza,
$\mathbb{N}_0$	–	a nemnegatív egészek halmaza,
$\mathbb{Z}$	–	az egész számok halmaza,
$\mathbb{L}$	–	a logikai értékek halmaza,
$\emptyset$	–	az üres halmaz.

Szokás a természetes számok halmazába a nullát is beleérteni, de ebben a könyvben erre külön jelölést ($\mathbb{N}_0$) használunk.

A halmazokat gyakran vagy az elemeik felsorolásával

$$\mathbb{L} ::= \{igaz, hamis\},$$

vagy egy tulajdonság megfogalmazásával

$$[a..b] ::= \{x \in \mathbb{Z} \mid x \geq a \text{ és } x \leq b\}$$

adjuk meg.

Természetesen használni fogjuk a matematikában megszokott halmazelméleti műveleteket

$\cup$	–	unió,
$\cap$	–	metszet,
$\setminus$	–	különbség,

és relációkat is

$\in$	–	eleme,
$\subseteq$	–	részhalmaza,
$\subset$	–	valódi része.

Egy H halmaz számosságát $|H|$ jelöli, mivel ebben a könyvben legfeljebb megszámlálható halmazokkal foglalkozunk, azt hogy egy H halmaz véges, néha így is írjuk, $|H| < \infty$.

Egy H halmaz részhalmazainak halmazát, azaz a hatványhalmazát $\wp(H)$val jelöljük.

1.2. Sorozatok

Ha A egy adott halmaz, akkor az $\alpha =< \alpha_1, \alpha_2, \cdots >, \alpha_i \in A$ egy A-beli véges, vagy végtelen sorozatot jelöl.

Az A-beli véges sorozatokat $\alpha =< \alpha_1, \alpha_2, \ldots, \alpha_n >, \alpha_i \in A$ alakban írhatjuk le. A véges sorozat hosszát $|\alpha|$ jelöli.

Az A-beli véges sorozatok halmazát A^*-gal, a végtelen sorozatok halmazát A^∞-nel jelöljük. Az előző két halmaz uniójaként előálló A-beli véges, vagy végtelen sorozatok halmazát A^{**}-gal jelöljük.

Egy $\alpha \in A^{**}$ sorozat értelmezési tartományát $\mathcal{D}_\alpha$-val jelöljük, és a következő halmazt értjük rajta:

$$\mathcal{D}_\alpha ::= \begin{cases} [1..|\alpha|], & \text{ha } \alpha \in A^* \\ \mathbb{N}, & \text{ha } \alpha \in A^\infty \end{cases}$$

Legyenek $\alpha^1, \alpha^2, \ldots, \alpha^{n-1} \in A^*$ és $\alpha^n \in A^{**}$. Ekkor azt a sorozatot, amit az $\alpha^1, \alpha^2, \ldots, \alpha^{n-1}, \alpha^n$ sorozatok egymás után írásával kapunk, a fenti sorozatok *konkatenációjának* nevezzük, és $kon(\alpha^1, \alpha^2, \ldots, \alpha^{n-1}, \alpha^n)$-nel jelöljük.

Egy A^{**}-beli sorozat *redukáltjának* nevezzük azt a sorozatot, amit úgy kapunk, hogy az eredeti sorozat minden azonos elemekből álló véges részsorozatát a részsorozat egyetlen elemével helyettesítjük. Egy $\alpha \in A^{**}$ sorozat redukáltját $red(\alpha)$-ával jelöljük.

Bevezetjük még a τ függvényt, ami egy véges sorozathoz hozzárendeli annak utolsó elemét: $\tau : A^* \to A$, $\forall \alpha \in A^*$:

$$\tau(\alpha) ::= \alpha_{|\alpha|}.$$

1.3. Relációk

Legyenek A és B tetszőleges halmazok, ekkor az

$$A \times B ::= \{(a, b) \mid a \in A \text{ és } b \in B\}$$

az A és B halmazok *direktszorzata*.

Az $R \subseteq A \times B$ halmazt *bináris relációnak* nevezzük. A továbbiakban, ha nem okoz félreértést a bináris jelzőt elhagyjuk.

A reláció *értelmezési tartománya*:

$$\mathcal{D}_R ::= \{a \in A \mid \exists b \in B : (a, b) \in R\},$$

a reláció *értékkészlete*:

$$\mathcal{R}_R ::= \{b \in B \mid \exists a \in A : (a, b) \in R\},$$

a reláció *értéke* egy $a \in A$ helyen, vagy a R szerinti *képe*:

$$R(a) ::= \{b \in B \mid (a, b) \in R\},$$

egy $H \subseteq A$ halmaz R szerinti *képe* a halmazt alkotó elemek képeinek uniója, azaz

$$R(H) ::= \bigcup_{h \in H} R(h).$$

Az $R \subseteq A \times B$ relációt felfoghatjuk egy leképezésnek, megfeleltetésnek is az A és a B halmaz elemei között. Az értelmezési tartomány jelenti azokat az A-beli elemeket, amikhez hozzárendelünk legalább egy B-beli elemet. Hasonlóan, az értékkészlet a legalább egy elelemhez hozzárendelt elemek halmaza. Egy $a \in A$ elemnek a képe, azoknak a B-beli elemeknek a halmaza, amiket a reláció hozzárendel az a-hoz. A H halmaképét az elemek képeinek uniójával definiáltuk, ez másképpen úgy fogalmazható, hogy a H képe azokból az elemekből áll, amik legalább egy H-beli elemhez hozzá vannak rendelve, azaz

$$R(H) = \{b \in B \mid \exists a \in H : (a, b) \in R\}.$$

Azt mondjuk, hogy egy reláció *determinisztikus*,vagy *parciális függvény*, ha

$$\forall a \in A : |R(a)| \leq 1.$$

Függvénynek nevezünk egy relációt akkor, ha

$$\forall a \in A : |R(a)| = 1.$$

Az A-ból B-be képező f függvényt általában $f : A \to B$-vel, a parciális függvényt $f \in A \to B$-vel jelöljük. E jelölés használata esetén $f(a)$ nem egy egyelmű halmazt, hanem annak elemét jelenti.

Legyen $R \subseteq A \times B$. Ekkor az $R^{(-1)}$ reláció az R *inverze*, ha

$$R^{(-1)} ::= \{(b, a) \in B \times A \mid (a, b) \in R\}.$$

Legyen $H \subseteq B$ tetszőleges halmaz. Ekkor az inverz reláció szerinti képét, $R^{(-1)}(H)$-t, a H halmaz R reláció szerinti *inverz képének* nevezzük. A definíciókból látszik, hogy

$$R^{(-1)}(H) = \{a \in A \mid R(a) \cap H \neq \emptyset\}.$$

Fontos megkülönböztetni az inverz képet a H halmaz R reláció szerinti *őskép*étől, aminek a definíciója a következő:

$$R^{-1}(H) ::= \{a \in \mathcal{D}_R \mid R(a) \subseteq H\}.$$

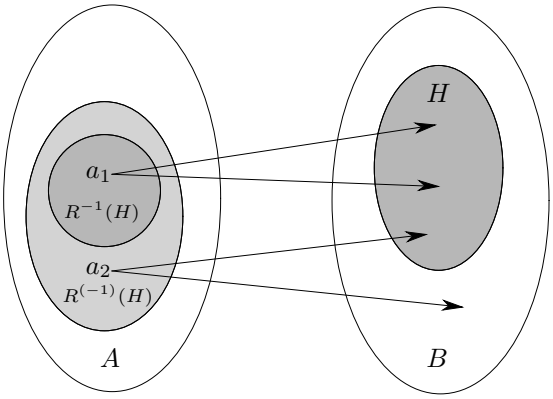
Az őskép általában nem egyezik meg az inverz képpel, de minmindigsze annak. A két kép kapcsolatát mutatja az 1.1 ábra. Megjegyezzük azonban, hogy függvény, illetve parciális függvény esetén a két kép megegyezik.

Legyen $P \subseteq A \times B$ és $Q \subseteq A \times B$. Ha $P \subseteq Q$ P-t Q *leszűkítés*ének nevezzük.

Ha $R \subseteq A \times B$ és $H \subseteq A$

$$R_{|H} ::= R \cap (H \times B)$$

R egy leszűkítése, amit úgy is nevezünk, hogy R *megszorítása* H-ra.



1.1. ábra. Inverz kép és őskép

1.3.1. Műveletek

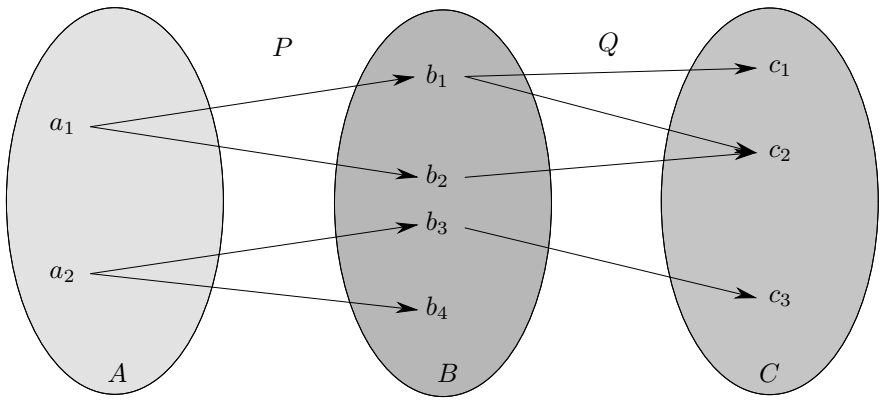
A relációk között értelmezünk műveleteket is. Legyen $P \subseteq A \times B$ és $Q \subseteq B \times C$. Ekkor az $R \subseteq A \times C$ relációt a P és Q relációk *kompozíciójának* nevezzük, ha

$$R = Q \circ P ::= \{(a, c) \in A \times C \mid \exists b \in B : (a, b) \in P \text{ és } (b, c) \in Q\}.$$

Az $S \subseteq A \times C$ relációt a P és Q relációk *szigorú kompozíciójának* nevezzük, ha

$$S = Q \odot P ::= \{(a, c) \in A \times C \mid \exists b \in B : (a, b) \in P \text{ és } (b, c) \in Q \text{ és } P(a) \subseteq \mathcal{D}_Q\}.$$

Mint az az 1.2 ábrán is látható, a kompozíció és a szigorú kompozíció általában nem egyezik meg, $(a_2, c_3) \in Q \circ P$, de nem eleme $Q \odot P$-nek. Könnyű belátni, a szigorú kompozíció minmindigsze a kompozíciónak. Azt sem nehéz belátni, hogy ha a két reláció közül legalább az egyik függvény, vagy ha az első parciális függvény, a kétféle kompozíció megegyezik.



1.2. ábra. Kompozíció és szigorú kompozíció

Ha $R \subseteq A \times A$ azt mondjuk, hogy R *homogén* reláció. A homogén reláció önmagával komponálható. Ennek alapján defi niáljuk R *hatvány*ait:

$$R^0 ::= \{(a, a) \mid a \in A\},$$

és ha $n \in \mathbb{N}$

$$R^n ::= R \circ R^{n-1}.$$

Az $\{(a,a) \mid a \in A\}$ relációt *identitás* relációnak is nevezzük és $id*_A$-val jelöljük.

Az $R \subseteq A \times A$ reláció *lezártja* az az $\overline{R} \subseteq A \times A$ reláció, amelyre:

(1) $\mathcal{D}_{\overline{R}} ::= \{a \in A \mid \nexists \alpha \in A^\infty : \alpha_1 \in R(a) \text{ és } \forall i \in \mathbb{N} : \alpha_{i+1} \in R(\alpha_i)\}$ és

(2) $\forall a \in \mathcal{D}_{\overline{R}} : \overline{R}(a) ::= \{b \in A \mid \exists k \in \mathbb{N}_0 : b \in R^k(a) \text{ és } b \notin \mathcal{D}_R\}$.

A lezárt tehát olyan pontokban van értelmezve, amelyekből kiindulva a relációt nem lehet végtelen sokszor egymás után alkalmazni, és ezekhez a pontokhoz olyan pontokat rendel, amelyeket úgy kapunk, hogy a reláció véges sokszori alkalmazásával kikerülünk az eredeti reláció értelmezési tartományából. Tehát $\mathcal{D}_R \cap \mathcal{R}_{\overline{R}} = \emptyset$ mindig teljesül és $\forall a \notin \mathcal{D}_R$-ra $a \in \mathcal{D}_{\overline{R}}$ és $R(a) = \{a\}$. Felhívjuk a figyelmet arra, hogy ez a definíció nem egyezik meg azzal, amit *tranzitív lezártnak* szoktak nevezni.

Az $R \subseteq A \times A$ reláció *korlátos lezártja* az az $\overline{\overline{R}} \subseteq A \times A$ reláció, amelyre:

(1) $\mathcal{D}_{\overline{\overline{R}}} ::= \{a \in A \mid \exists k_a \in \mathbb{N}_0 : R^{k_a}(a) = \emptyset\}$ és

(2) $\forall a \in \mathcal{D}_{\overline{\overline{R}}} : \ \overline{\overline{R}}(a) ::= \overline{R}(a)$.

Vegyük észre, hogy egy reláció lezártja, és korlátos lezártja különbözhet. A definíciókból látható, hogy ez a különbözőség csak az értelmezési tartományok között jelentkezhet. Ennek azonban szükséges feltétele, hogy egy alkalmas pontból kiindulva a relációt ne lehessen végtelen sokszor alkalmazni, de a véges sokszori alkalmazások hosszára ne tudjunk korlátot mondani. Természetesen ez csak akkor lehetséges, ha végtelen sok véges alkalmazás-sorozatot tudunk a pontból indítani. Nézzünk erre egy egyszerű példát: legyen $R \subseteq \mathbb{Z} \times \mathbb{Z}$, és

$$R(a) = \begin{cases} \{a-1\}, & \text{ha } a > 0 \\ \mathbb{N}, & \text{ha } a < 0 \end{cases}$$

Ekkor $\mathbb{Z} = \mathcal{D}_{\overline{R}} \neq \mathcal{D}_{\overline{\overline{R}}} = \mathbb{N}_0$.

1.3.2. Logikai relációk

Az $R \subseteq A \times \mathbb{L}$ típusú relációkat – ahol A tetszőleges halmaz –, logikai relációknak nevezzük. A logikai relációkra bevezetünk néhány jelölést:

Legyen $R \subseteq A \times \mathbb{L}$. Ekkor az R *igazsághalmaza:*

$$\lceil R \rceil ::= R^{-1}(\{igaz\}),$$

gyenge igazsághalmaza:

$$\lfloor R \rfloor ::= R^{(-1)}(\{igaz\}).$$

Ha R függvény, akkor az igazsághalmaz és gyenge igazsághalmaz megegyezik.

Legyen $H \subseteq A$, azt $A \to \mathbb{L}$ függvényt aminek az igazsághalmaza H a H halmaz karakterisztikus függvényének nevezzük és $\mathcal{P}(H)$-val jelöljük A fenti definíciókból következik, hogy tulajdonképpen mindegy, hogy egy halmaz részhalmazairól, vagy a halmazon értelmezett logikai függvényekről (állításokról) beszélünk, hiszen ezen fogalmak kölcsönösen egyértelműen megfelelnek egymásnak.

Gyakran fogjuk használni az azonosan igaz és az azonosan hamis függvényeket: $Igaz_A : A \to \mathbb{L}$ és $\lceil Igaz_A \rceil = A$, $Hamis_A : A \to \mathbb{L}$ és $\lceil Hamis_A \rceil = \emptyset$. Ha nem okoz félreértést az A indexet nem írjuk ki.

Legyen $R \subseteq A \times A$ és $\pi : A \to \mathbb{L}$. Az

$$R|\pi = R|_{\lceil \pi \rceil} \cup \{(a,a) \in A \times A \mid a \in \lceil \pi \rceil \setminus \mathcal{D}_R\}.$$

relációt feltételes relációnak nevezzük, ami a reláció leszűkítése és kiterjesztése a feltétel igazsághalmazára, azaz $\mathcal{D}_{R|\pi} = \lceil \pi \rceil$.

A továbbiakban egy feltételes reláció lezártját, illetve korlátos lezártját a reláció *feltételre vonatkozó lezártjának*, illetve *feltételre vonatkozó korlátos lezártjának* fogjuk nevezni.

1.4. Direktszorzat

Legyenek $A_i, i \in I$ tetszőleges halmazok, és $X = \{x \mid x : I \to \cup_{i \in I} A_i\}$ azaz X az I-t az A_i-k uniójába képező függvények halmaza. Ekkor az

$$A = \mathop{\times}_{i \in I} A_i ::= \{x \in X \mid \forall i \in I : x(i) \in A_i\}$$

halmazt az $A_i, i \in I$ halmazok *direktszorzatának* nevezzük.

Az I halmaz gyakran az $[1..n]$ intervallum, ekkor az

$$A = \mathop{\times}_{i=1}^{n} A_i$$

jelölést használjuk. Ebben az esetben azt mondhatjuk, hogy a direktszorzat elemei rendezett n-esek. Az általános esetben is a direktszorzat elemeit gyakran mint rendezett n-eseket adjuk meg, feltételezve, hogy egyértelműen el tudjuk dönteni, hanyadik komponens melyik $i \in I$-hez tartozik.

Ha $J \subseteq I$ és $K = I \setminus J$ a

$$B = \mathop{\times}_{j \in J} A_j$$

direktszorzat *altere* és a

$$B' = \mathop{\times}_{k \in K} A_k$$

direktszorzat *kiegészítő altere* A-nak.

A $pr_B : A \to B$ függvényt *projekciónak* nevezzük, ha

$$\forall a \in A : pr_B(a) = a|_J.$$

Ha $J = \{j\}$ a pr_B függvényt j-edik projekciónak, vagy j *változónak* is nevezzük.

Értelemezzük a projekciót párokra, sorozatokra és halmazokra is:
$pr_B((a_1, a_2)) ::= (pr_B(a_1), pr_B(a_2))$
$pr_B(\langle a_1, a_2, \dots \rangle) ::= \langle pr_B(a_1), pr_B(a_2), \dots \rangle$
$pr_B(\{a_1, a_2, \dots\}) ::= \{pr_B(a_1)\} \cup \{pr_B(a_2)\} \cup \dots$

azaz a párok vetülete a komponensek vetületéből álló pár, a sorozat vetülete egy, az eredetivel azonos hosszúságú sorozat, aminek elemei rendre az eredeti sorozat elemeinek vetületei. A halmaz vetülete halmaz, de lehet, hogy a vetület halmaz elemeinek száma kisebb mint az eredetei volt, egy végtelen halmaz vetülete lehet véges is.

1.5. Függvényterek

Ha $f, g : A \to B$ függvények és $\oplus$ egy művelet B-n, azaz $\oplus : B \times B \to B$, akkor definiálhatjuk az $f \oplus g : A \to B$ függvényt is, úgy, hogy

$$\forall a \in A : f \oplus g(a) = f(a) \oplus g(a).$$

Parciális függvények esetén

$$\mathcal{D}_{f \oplus g} = \{a \in A \mid a \in \mathcal{D}_f \text{ és } a \in \mathcal{D}_g \text{ és } (f(a), g(a)) \in \mathcal{D}_\oplus\}.$$

Az f, g függvények és egy $\sim \subseteq B \times B$ reláció logikai függvényeket definiálnak: $f \sim g : A \to \mathbb{L}$ és

$$\forall a \in A : f \sim g(a) = \begin{cases} igaz & \text{ha } a \in \mathcal{D}_f \text{ és } a \in \mathcal{D}_g \text{ és } (f(a), g(a)) \in \sim, \\ hamis & \text{egyébként.} \end{cases}$$

Az így kapott logikai függvényekre a fentebb definiált módon alkalmazhatjuk a szokásos logikai műveleteket: $\wedge, \vee, \to, \equiv, \neg$. Megjegyezzük, hogy a definícióból rögtön adódnak a következő összefüggések. Legyen $f, g : A \to \mathbb{L}$, ekkor:

$$\lceil f \wedge g \rceil = \lceil f \rceil \cap \lceil g \rceil,$$
$$\lceil f \vee g \rceil = \lceil f \rceil \cup \lceil g \rceil,$$
$$\lceil \neg f \rceil = A \setminus \lceil f \rceil.$$

Az $f = g$ jelölést az $f \equiv g$ helyett használjuk és

$$\lceil f = g \rceil = \lceil f \rceil \cap \lceil g \rceil \cup \lceil \neg f \rceil \cap \lceil \neg g \rceil.$$

Az implikáció jelölésére gyakran használják$\Rightarrow$ jelet $\to$ helyett, mi az előbbit a "következik" reláció jelölésére használjuk, azaz

$$f \Rightarrow g \Leftrightarrow \lceil f \rceil \subseteq \lceil g \rceil.$$

A $\forall$ és a $\exists$ jeleket eddig is használatuk mint a "minden" és "létezik" szavak rrövidítéseit A fenti logikai kifejezések esetén $\forall i \in I : \cdots$ jelentése $\bigwedge_{i \in I} \cdots$ és $\exists i \in I : \cdots$ jelentése $\bigvee_{i \in I} \cdots$. Ha $I = \emptyset$, $\bigwedge_{i \in I} \cdots$ azonosan igaz, $\bigvee_{i \in I} \cdots$ pedig azonosan hamis.

1.6. Példák

1.1. példa: Írjuk fel az $A \times B$, $A \times C$, $(A \times B) \times C$, és $A \times B \times C$ halmazok elemeit, ha $A = \{0, 1\}, B = \{1, 2, 3\}, C = \{p, q\}$!
Megoldás:

$$\begin{aligned} A \times B =& \{(0,1),(0,2),(0,3),(1,1),(1,2),(1,3)\}, \\ A \times C =& \{(0,p),(0,q),(1,p),(1,q)\}, \\ (A \times B) \times C =& \{((0,1),p),((0,2),p),((0,3),p),((1,1),p),((1,2),p),((1,3),p), \\ & ((0,1),q),((0,2),q),((0,3),q),((1,1),q),((1,2),q),((1,3),q)\}, \\ A \times B \times C =& \{(0,1,p),(0,2,p),(0,3,p),(1,1,p),(1,2,p),(1,3,p), \\ & (0,1,q),(0,2,q),(0,3,q),(1,1,q),(1,2,q),(1,3,q)i\}. \end{aligned}$$

1.2. példa: Legyen $R \subseteq \{1, 2, 3, 4, 5\} \times \{1, 2, 3, 4, 5\}$.

$$R = \{(1,2),(1,4),(2,1),(3,4),(3,3),(3,5),(4,5)\}.$$

a) Mi a reláció értelmezési tartománya és értékkészlete?
b) Determinisztikus-e, illetve függvény-e a reláció?
c) Mi $R\,0., 2., (-1).$ hatványa?
d) Mi a $\{4, 5\}$ halmaz inverz képe, illetve ősképe?

Megoldás:

a) $\mathcal{D}_R = \{1, 2, 3, 4\}$,
$\mathcal{R}_R = \{1, 2, 3, 4, 5\}$.

b) A reláció nem determinisztikus, ugyanis pl. $|R(1)| = 2$! Mivel a reláció nem determinisztikus, függvény sem lehet.

c) A reláció $0.$ hatványa az identikus leképezés, azaz:

$$R^0 = \{(1, 1), (2, 2), (3, 3), (4, 4), (5, 5)\}.$$

Mivel $R^2 = R \circ R$, azt kell megvizsgálnunk, hogy mely pontokból hogyan lehet a relációt egymás után kétszer alkalmazni:

$$\begin{array}{lcl} (1,2) & \longrightarrow & (2,1) \\ (1,4) & \longrightarrow & (4,5) \\ (2,1) & \longrightarrow & (1,2) \\ (2,1) & \longrightarrow & (1,4) \\ (3,4) & \longrightarrow & (4,5) \\ (3,3) & \longrightarrow & (3,4) \\ (3,3) & \longrightarrow & (3,3) \\ (3,3) & \longrightarrow & (3,5) \end{array}$$

A fenti táblázat alapján:

$$R^2 = \{(1, 1), (1, 5), (2, 2), (2, 4), (3, 5), (3, 4), (3, 3)\}.$$

$R^{(-1)}$ a reláció inverzének defi níciója alapján:

$$R = \{(2, 1), (4, 1), (1, 2), (4, 3), (3, 3), (5, 3), (5, 4)\}.$$

d) Írjuk fel, hogy mit rendel a reláció az értelmezési tartomány egyes pontjaihoz:

$$\begin{array}{lcl} R(1) & = & \{2, 4\} \\ R(2) & = & \{1\} \\ R(3) & = & \{3, 4, 5\} \\ R(4) & = & \{5\} \end{array}$$

Az inverz kép defi níciója alapján:

$$R^{(-1)}(\{4, 5\}) = \{1, 3, 4\}.$$

Az őskép defi níciója alapján:

$$R^{-1}(\{4, 5\}) = \{4\}.$$

1.3. példa: Megadható-e valamilyen összefüggés egy H halmaz inverz képének képe, és a H halmaz között?
Megoldás: Legyen $R \subseteq A \times B$, $H \subseteq B$. Ekkor

$$\begin{aligned} R(R^{(-1)}(H)) &= R(\{a \in A \mid R(a) \cap H \neq \emptyset\}) = \\ &= \bigcup_{R(a)\cap H\neq\emptyset} R(a). \end{aligned}$$

Vegyük észre, hogy általános esetben nem tudunk mondani semmit a két halmaz viszonyáról, ugyanis

1. ha $H \not\subseteq \mathcal{R}_R$, akkor $H \not\subseteq R(R^{(-1)}(H))$ és
2. ha $\exists a \in R^{(-1)}(H) : R(a) \not\subseteq H$, akkor $R(R^{(-1)}(H)) \not\subseteq H$.

Tekintsük e fenti esetet egy egyszerű számpéldán: Legyen $A = B = \{1,2,3\}$, $R = \{(1,1),(1,2)\}$. Ekkor $H = \{2,3\}$ esetén $R(R^{(-1)}(H)) = \{1,2\}$, azaz egyik irányú tartalmazás sem áll fenn.

1.4. példa: Legyen $R \subseteq A \times B$, $P, Q \subseteq B$. Hogyan lehetne jellemezni az $R^{-1}(P \cup Q)$ és az $R^{-1}(P \cap Q)$ halmazt az $R^{-1}(P)$ és $R^{-1}(Q)$ halmaz segítségével?
Megoldás:

$$\begin{aligned} R^{-1}(P \cup Q) &= \{a \in \mathcal{D}_R \mid R(a) \subseteq (P \cup Q)\} = \\ &\supseteq \{a \in \mathcal{D}_R \mid R(a) \subseteq P\} \cup \{a \in \mathcal{D}_R \mid R(a) \subseteq Q\}. \end{aligned}$$

A másik irányú tartalmazás azonban nem áll fenn, ugyanis lehet olyan $a \in \mathcal{D}_R$ amelyre

$$R(a) \not\subseteq P, \text{ és } R(a) \not\subseteq Q, \text{ de } R(a) \subseteq P \cup Q.$$

Nézzük ezt egy számpéldán: Legyen $A = B = \{1,2\}$, $R = \{(1,1),(1,2)\}$, $P = \{1\}$, $Q = \{2\}$. Ekkor $R^{-1}(P)$ és $R^{-1}(Q)$ üres, de $R^{-1}(P \cup Q) = \{1\}$.
Vizsgáljuk most meg a metszetet!

$$\begin{aligned} R^{-1}(P \cap Q) &= \{a \in \mathcal{D}_R \mid R(a) \subseteq (P \cap Q)\} = \\ &= \{a \in \mathcal{D}_R \mid R(a) \subseteq P\} \cap \{a \in \mathcal{D}_R \mid R(a) \subseteq Q\} = \\ &= R^{-1}(P) \cap R^{-1}(Q). \end{aligned}$$

Tehát bebizonyítottuk, hogy két tetszőleges halmaz metszetének ősképe egyenlő a két halmaz ősképének metszetével.

1.5. példa: Legyenek $F \subseteq A \times B, G \subseteq B \times C$. Igaz-e, hogy

$$(G \circ F)^{(-1)} = F^{(-1)} \circ G^{(-1)}?$$

Megoldás:

$$\begin{aligned} (G \circ F)^{(-1)} &= \{(c,a) \in C \times A \mid \exists b \in B : (a,b) \in F \text{ és } (b,c) \in G\} = \\ &= \{(c,a) \in C \times A \mid \exists b \in B : (b,a) \in F^{(-1)} \text{ és } (c,b) \in G^{(-1)}\} = \\ &= F^{(-1)} \circ G^{(-1)}. \end{aligned}$$

1.6. példa: Legyenek $F \subseteq A \times B, G \subseteq B \times C$. Igaz-e, hogy

$$\forall Y \subseteq B : (G \odot F)^{-1}(Y) = F^{-1}(G^{-1}(Y))?$$

Megoldás:A szigorú kompozíció definíciójából azonnalal adódik, hogy ha $a \in A$ és $a \in \mathcal{D}_{G\odot F}$, akkor $G \odot F(a) = G(F(a))$, ezt felhasználva:

$$\begin{aligned}(G \odot F)^{-1}(Y) &= i\{a \in A \mid a \in \mathcal{D}_{F\odot G} \text{ és } (G \odot F)(a) \subseteq Y\} \\ &= \{a \in A \mid a \in \mathcal{D}_F \text{ és } F(a) \subseteq \mathcal{D}_G \text{ és } G(F(a)) \subseteq Y\} \\ &= \{a \in A \mid a \in \mathcal{D}_F \text{ és } F(a) \subseteq \{b \in B \mid b \in \mathcal{D}_G \text{ és } G(b) \subseteq Y\}\} \\ &= F^{-1}(G^{-1}(Y)).\end{aligned}$$

1.7. példa: $W = N_1 \times N_2 \times N_3$. $\alpha \in W^{**}$, ahol $N_i = \mathbb{N}$ $(i = 1, 2, 3)$. $\alpha_1 = (1, 1, 1)$. Az α sorozat további elemeit úgy kapjuk meg, hogy a pontok koordinátáit az első koordinátával kezdve ciklikusan 1-gyel növeljük. $red(pr_{N_1\times N_3}(\alpha)) =$?
Megoldás: Írjuk fel először a sorozat első néhány tagját:

$$\alpha =< (1, 1, 1), (2, 1, 1), (2, 2, 1), (2, 2, 2), (3, 2, 2), (3, 3, 2) \cdots >$$

Az α sorozat projekciója $N_1 \times N_3$-ra:

$$pr_{N_1\times N_3}(\alpha) =< (1, 1), (2, 1), (2, 1), (2, 2), (3, 2), (3, 2) \cdots >$$

A fenti sorozat redukáltja:

$$red(pr_{N_1\times N_3}(\alpha)) =< (1, 1), (2, 1), (2, 2), (3, 2) \cdots >$$

A fentiekből jól látható, hogy a redukció pontosan azokat az elemeket hagyja ki a sorozatból, amelyekben a növelés a második komponensben történt, így az eredménysorozat elemeit is a koordináták ciklikus eggyel növelésével kapjuk meg, az $(1, 1)$ pontból kiindulva.

1.8. példa: Legyen $A = \{1, 2, 3, 4, 5\}$ és $R \subseteq A\times A$. $R = \{(1, 2), (1, 4), (2, 1), (3, 4), (3, 3), (3, 5), (4, 5)\}$. Mi lesz R lezártja és korlátos lezártja?
Megoldás: Mivel $\mathcal{D}_R = \{1, 2, 3, 4\}$, azt kell csak megvizsgálni, hogy honnan jutunk el biztosan a reláció ismételt alkalmazásával 5-be. $R(1) = \{2, 4\}$ és $R(2) = \{1\}$, ezért $1, 2 \notin \mathcal{D}_{\overline{R}}$, a 3 sem, mert a 3-ból akárhányszor eljuthatunk 3-ba, 4-ből egy lépésben, 5-ből nulla lépésben jutunk 5-be. Tehát $\overline{R} = \{4, 5\}$ és $\overline{\overline{R}} = \overline{R}$.

1.9. példa:
$A = \{1, 2, 3, 4, 5\}, R \subseteq A \times A$. $R = \{(1, 2), (1, 5), (2, 5), (3, 2), (3, 4), (5, 2)\}$. $\lceil\pi\rceil = \{1, 2, 3, 4\}$. írjuk fel a reláció feltételre vonatkozó lezártját!
Megoldás:$R|\pi = \{(1, 2), (1, 5), (2, 5), (3, 2), (3, 4), (4, 4)\}$. Az $(5, 2)$ kimaradt a szűkítés miatt, a $(4, 4)$ pedig bekerült a bővítés miatt. $\overline{R|\pi} = \{(1, 5), (2, 5), (5, 5)\}$.

1.10. példa: Van-e olyan nem üres reláció és π feltétel, hogy a reláció lezártja üres halmaz, és a π feltételre vonatkozó lezártja azonos a relációval?
Megoldás: Legyen A tetszőleges halmaz. Nyilván $\overline{id_A} = \emptyset$. Ha $\pi = Hamis$, $id_A|\pi = \emptyset$, aminek a lezártja id_A.

1.11. példa: $R \subseteq \mathbb{N}_0 \times \mathbb{N}_0$.

$$R(a) = \begin{cases} \{a - 2\}, & \text{ha } a > 1 \\ \{2^k \mid k \in \mathbb{N}\}, & \text{ha } a = 1 \end{cases}$$

Mi az R reláció lezártja és korlátos lezártja?

Megoldás: $\mathcal{D}_R = \mathbb{N}$. Miden a-hoz, ha a páros a reláció $a/2$ lépésben hozzárendeli a 0-t és csak azt, ha pedig páratlan, az 1-et, aminek a képe az összes páros kettőhatvány. A kettőhatványokból, mivel mind páros, az R ismételt alkalmazása 0-ba vezet. Tehát $\mathcal{D}_{\overline{\overline{R}}} = \mathbb{N}_0$, és természetesen $\forall a \in \mathbb{N}_0 : \overline{R(a)} = 0$.

Mivel nincs felső korlátja a kettőhatványoknak, ezért $\mathcal{D}_{\overline{\overline{R}}}$ nem tartalmazza a páratlan számokat.

Megjegyezzük, hogy ha a feladatban $\{2^k \mid k \in \mathbb{N}\}$ helyett $\{2^k \mid k \in \mathbb{N}_0\}$ szerepelne, $\mathcal{D}_{\overline{R}}$ sem tartalmazná a páratlan számokat.

1.7. Feladatok

1.1. Milyen összefüggés van egy H halmaz R relációra vonatkozó inverz képe és ősképe között? És ha R függvény?

1.2. $R = \{((x, y), (x + y, y)) \mid x, y \in \mathbb{N}\}$. Mi a $H = \{(a, b) \mid a, b \in \mathbb{N} \wedge a + b < 5\}$ halmaz inverz képe, ill. ősképe?

1.3. $R = \{((x, y), (x + y, y)) \mid x, y \in \mathbb{N}\} \cup \{((x, y), (x - y, y)) \mid x, y \in \mathbb{N}\}$. Mi a $H = \{(a, b) \mid a, b \in \mathbb{N} \wedge a + b < 5\}$ halmaz inverz képe, ill. ősképe?

1.4. $R = \{((x, y), (f(x, y), y)) \mid x, y \in \mathbb{N}\}$, ahol $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. Mi a $H = \{(a, b) \mid a, b \in \mathbb{N} \wedge a + b < 5\}$ halmaz ősképe illetve inverz képe?

1.5. $R \subseteq A \times B, Q \subseteq B$. Van-e valamilyen összefüggés az $R^{-1}(B \setminus Q)$ halmaz és az $A \setminus (R^{-1}(Q))$ halmaz között?

1.6. Készíts olyan nem üres relációt, amelyre igaz, hogy értékkészlete minden valódi részhalmazának ősképe üres halmaz!

1.7. Legyen $A = \{1, 2, 3, 4, 5\}$, $R \subseteq A \times A$, $R = \{(1, 2), (1, 4), (2, 1), (3, 4), (3, 3), (3, 5), (4, 5)\}$, $f \subseteq A \times \mathbb{L}$ és $f = \{(1, i), (2, i), (3, i), (4, h), (5, i)\}$. Mi f, illetve $(f \circ R)$ igazsághalmaza?

1.8. $R, Q \subseteq A \times A$. Igaz-e, hogy $(R \odot Q)^{(-1)} = Q^{(-1)} \circ R^{(-1)}$?

1.9. $R \subseteq A \times A$. Igaz-e, hogy $(R^{(-1)})^2 = (R^2)^{(-1)}$?

1.10. $R \subseteq A \times A$. Igaz-e, hogy $\forall H \subseteq A : R^{-1}(R^{-1}(H)) = (R^2)^{-1}(H)$?

1.11. $P, Q \subseteq \mathbb{N} \times \mathbb{N}$. $Q = \{(a, b) \mid 2|a$ és $b|a$ és $prim(b)\}$.

a) $P = \{(a, b) \mid b|a$ és $b \neq 1$ és $b \neq a\}$

b) $P = \{(a, b) \mid b|a\}$

Add meg a $Q^{(-1)}, Q \circ P$ és $Q \odot P$-t relációt!

1.12. Legyen $Q, R, S \subseteq A \times A$, és vezessük be az alábbi jelölést: ha $X \subseteq A \times A$ tetszőleges reláció, akkor X komplementere:

$$\widehat{X} = \{(a, b) \in A \times A \mid (a, b) \notin X\}.$$

Igaz-e, hogy

$$Q \odot R \subseteq S \Longleftrightarrow Q^{(-1)} \odot \widehat{S} \subseteq \widehat{R}?$$

Igaz-e a fenti állítás nem-szigorú kompozíció esetén?

1.13. Legyen $Q, R, S \subseteq A \times A$. Igaz-e, hogy

$$R \subseteq S \quad \Rightarrow \quad R \odot Q \subseteq S \odot Q,$$
$$R \subseteq S \quad \Rightarrow \quad Q \odot R \subseteq Q \odot S?$$

1.14. Legyen R és Q két reláció a természetes számok halmazán! R egy természetes számhoz rendeli önmagát és a kétszeresét, Q egy páros természetes számhoz a felét.

a) Írd fel a két relációt, és add meg az értelmezési tartományukat!

b) Írd fel az R reláció k. hatványát ($k \geq 1$) és ennek az értelmezési tartományát!

c) Írd fel a $Q \circ R$ relációt és az értelmezési tartományát!

d) $F = Q \odot R$! Írd fel az F relációt és az értelmezési tartományát!

1.15. $P \subseteq \mathbb{N}_0 \times \mathbb{N}_0$. $P = \{(a, b) \mid \ b|a$ és $b \neq 1$ és $b \neq a\}$. Mi lesz P lezártja?

1.16. Mutassunk példát olyan relációra, aminek lezártja és korlátos lezártja különböző!

1.17. $R \subseteq \mathbb{N} \times \mathbb{N}$. $R = \{(a, b) \mid \ b|a$ és $b \neq 1$ és $b \neq a\}$. $\lceil \pi \rceil = \{x \mid x$ kettőhatvány $\}$. írjuk fel az $R|_\pi$ relációt, lezártját és korlátos lezártját!

1.18. Adjunk példát olyan nem üres relációra, amelynek lezártja üres halmaz és van olyan π feltétel, hogy a reláció feltételre vonatkozó lezártjának értelmezési tartománya megegyezik az eredeti reláció értelmezési tartományával!

1.19. $R \subseteq A \times A$. Tegyük fel, hogy az R értelmezési tartománya egyenlő az R értelmezési tartományának R-re vonatkozó ősképével. Mit mondhatunk R lezártjáról?

1.20. $R \subseteq \mathbb{N}_0 \times \mathbb{N}_0$.

$$R(a) = \begin{cases} \{a - 3\}, & \text{ha } a > 2 \\ \{3 * k \mid k \in \mathbb{N}\}, & \text{ha } a = 1 \end{cases}$$

Mi az R reláció lezártja és korlátos lezártja?

1.21. $R \subseteq \mathbb{N} \times \mathbb{N}$. Az R reláció minden összetett számhoz a legnagyobb valódi osztóját rendeli. Legyen q

a) egy rögzített összetett természetes szám!

b) egy rögzített prímszám!

Legyen $P_q(a) = (\exists k \in \mathbb{N} \mid a = q^k)$! Mi lesz az R reláció P_q feltételre vonatkozó lezártjának értelmezési tartománya?

1.22. $R \subseteq \mathbb{N}_0 \times \mathbb{N}_0$.

$$R(x) = \begin{cases} \{b \mid \exists k \in \mathbb{N}_0 : \ b = 2 * k + 1\}, & \text{ha } x \neq 0 \text{ és } x \text{ páros} \\ \{x - 7\}, & \text{ha } x \geq 7 \text{ és } x \text{ páratlan} \\ \{0\}, & \text{ha } x = 1 \\ \{7\}, & \text{ha } x = 0 \end{cases}$$

Mi lesz R lezártja és korlátos lezártja?

1.23. R legyen a 21. feladatban adott reláció. $\pi(k) =$ (k páratlan szám). Add meg az $R|_\pi$ relációt, lezártját és korlátos lezártját!

1.24. Igazak-e az alábbi állítások?

a) Ha $a \in \mathcal{D}_{\overline{R}} \cap \mathcal{D}_{\overline{\overline{R}}}$, akkor $\overline{R}(a) = \overline{\overline{R}}(a)$.

b) $\mathcal{D}_{\overline{\overline{R}}} \subseteq \mathcal{D}_{\overline{R}}$.

* c) Ha az A halmaz véges és $R \subseteq A \times A$, akkor $\overline{R} = \overline{\overline{R}}$.

** d) Ha A megszámlálhatóan végtelen, $R \subseteq A \times A$, és

$$\forall a \in A : \ (\exists n(a) \in \mathbb{N}_0 : \ |R(a)| \leq n(a)) \Rightarrow \overline{R} = \overline{\overline{R}}.$$

1.25. Legyen $R \subseteq \mathbb{N}_0 \times \mathbb{N}_0$, R értelmezési tartománya $\mathbb{N}$!

$$R(x) = \begin{cases} \{b | b > 0 \text{ és } b < x \text{ és } 2|b\}, & \text{ha } x \text{ páratlan} \\ \{x - 1\}, & \text{ha } x \text{ páros} \end{cases}$$

$\pi(x) =$ (x páros természetes szám). Mi az R reláció π feltételre vonatkozó lezártja és korlátos lezártja?

1.26. Legfeljebb illetve legalább milyen hosszú egy m és egy n hosszúságú sorozat redukáltjának konkatenációja, illetve konkatenációjának redukáltja?

1.27. Igaz-e, hogy egy α sorozat redukáltjának projekciója ugyanolyan hosszú, mint az α sorozat redukáltja?

1.28. Igaz-e, hogy egy α sorozat projekciójának redukáltja ugyanolyan hosszú, mint az α sorozat redukáltja?

1.29. Legyen $A = N_1 \times N_2 \times N_3 \times N_4$, $B = N_4 \times N_1$, ahol $N_i = \mathbb{N}$ $(i = 1..4)$.

$$\begin{aligned} \alpha \quad = \quad & <(1,1,1,1),(1,2,1,1),(1,2,3,1),(1,2,3,4), \\ & (5,2,3,4),(5,7,3,4),(5,7,10,4),\cdots> \end{aligned}$$

a) $pr_B(\alpha) =$?

b) $red(pr_B(\alpha)) =$?

2. fejezet

A programozás alapfogalmai

Ebben a fejezetben a programozás legfontosabb alapfogalmai vezetjük be. Nagyon fontos szempont, hogy fi gyelmünket nem a programok tulajdonságainak vizsgálatára, hanem a programok előállítására fordítjuk. Ezért feltesszük a kérdést, miért is írunk programot? Az erre a kérdésre adott válasz meghatározza gondolkodásunk irányát. A válaszunk az, hogy azért, mert van valami megoldandó feladatunk, problémánk. Tehát a gondolkodásuk kiinduló pontja az, hogy kell lenni valaminek, amit feladatnak hívunk, s ez a feladat határozza meg az elérendő célt.

A gyakorlatban nagyon sokféle feladat van. Mi bennük a közös? Ez a kérdés is többféleképpen közelíthető meg. A mi megközelítésük szerint a feladat lényege az, hogy meghatározzuk, milyen állapotban vagyunk és milyen állapotba akarunk jutni. Az hogy mik az állapotok, a konkrét problémától függ. Például, ha egy autót akarunk egy hosszabb útra felkészíteni az állapotát jellemezheti az, hogy mennyi a tankban a benzin, mennyi az ablakmosó folyadék, mekkora a nyomás a kerekekben, működik-e az irányjelző és így tovább. A lényeg az, hogy van a rendszernek valahány jellemzője, ezen jellemzők lehetséges értékei egy-egy halmazt alkotnak. Egy ilyen halmaz állhat a mennyiséget kifejező számokból, lehet akár a { *működik, nem működik*} halmaz is.

Ha mindegyik jellemző lehetséges értékeinek halmazából választunk egy-egy értéket megkapjuk az autó egy lehetséges állapotát. Márcsak az hiányzik, hogy észrevegyük, a lehetséges állapotok halmaza matematikailag egy direktszorzat.

2.1. Az állapottér fogalma

Az elsőként bevezetendő absztrakt fogalom a fent említett lehetséges állapotok halmaza.

2.1. Definíció (ÁLLAPOTTÉR). *Legyen I egy véges halmaz és legyenek $A_i, i \in I$ tetszőleges véges vagy megszámlálható, nem üres halmazok. Ekkor az $A = \bigtimes_{i \in I} A_i$ halmazt* állapottérnek, *az A_i halmazokat pedig* típusértékhalmazoknak *nevezzük.*

Amikor egy modellt készítünk, el kell döntenünk, hogy a valóság mely részét kívánjuk modellezni, és melyek azok a jellemzők – és milyen értékeket vehetnek fel – amiket a modellünkben fi gyelembe akarunk venni.

Az állapottér fenti defi níciójában az egyes komponenseket tekintsük úgy, mint egyes jellemzők lehetséges értékeinek halmazát. A típusértékhalmaz elnevezés arra utal, hogy ezek a halmazok bizonyos közös tulajdonsággal rendelkező elemekből áll-

nak. A későbbiekben majd kitérünk arra is, hogy ez a közös tulajdonság mit is jelent. Mivel a jellemzők értékhalmaza lehet azonos, az állapottér komponensei között egy halmaz többször is szerepelhet.

Kikötöttük, hogy az állapottérnek csak véges sok komponense legyen. Lehetne általánosabb definíciót is adni, úgy ,hogy nem kötjük ki az I halmaz végességét, ekkor a fent definiált állapotteret az általánosított állapottér egy (véges)nézetének nevezzük.

Az, hogy a komponensek legfeljebb megszámlálhatók, azt is jelenti, hogy a komponensek között nem szerepelhet a pl. valós számok halmaza. Természetesen ettől még egy típusértékhalmaz tartalmazhatja akár $\sqrt{2}$-t is. Az $\{x \mid \exists n \in \mathbb{N} : x = \sqrt{n}\}$ lehet állapottér komponens.

2.2. A feladat

Az állapottér fogalmának segítségével könnyen megfogalmazhatjuk, hogy mit értünk feladaton. Azt kell megfogalmaznunk, hogy egy adott állapotból (azaz az állapottér egy eleméből, pontjából) milyen állapotba (azaz az állapottér mely pontjába) akarunk eljutni.

2.2. Definíció (FELADAT). Feladatnak *nevezzük az* $F \subseteq A \times A$ *relációt.*

A feladat fenti definíciója természetes módon adódik, abból, hogy a feladatot egy leképezésnek tekintjük az állapottéren, és az állapottér minden pontjára megmondjuk, hogy hova kell belőle eljutni, ha egyáltalán el kell jutni belőle valahova.

Az, hogy egy feladatnak mi lesz az állapottere, természetesen magától a feladattól függ, ám még a feladat ismeretében sem egyértelmű. Például egy pont síkbeli koordinátáit megadhatjuk derékszögűgű koordináta-rendszerben, de megadhatjuk polárkoordinátákkal is.

Mégis, az, hogy mit választunk állapottérnek, nagyon fontos, hiszen meghatározza, hogy a továbbiakban mit, és hogyan tudunk leírni. Ha túl kevés jellemzőt vizsgálunk – azaz az állapottér túl kevés komponensből áll – akkor lehetnek olyan fogalmak, amiket nem tudunk benne leírni, ha túl sok a komponens, akkor fölöslegesen túl bonyolult lesz a modell.

Tekintsük azt az egyszerű feladatot, hogy össze kelladni két természetes számot. Az állapotteret elég kézenfekvő módon három komponensűnek választhatjuk. A három komponens a két összeadandó és az összeg. Tehát $A = \mathbb{N} \times \mathbb{N} \times \mathbb{N}$, s a feladat

$$F = \{((a, b, c), (x, y, z)) \in A \times A \mid a + b = z\},$$

vagy

$$G = \{((a, b, c), (x, y, z)) \in A \times A \mid b = x \text{ és } c = y \text{ és } a + b = z\}.$$

A két feladat nem azonos bár mindkettő két természetes szám összegéről szól. A különbség köztük az, hogy az F feladat nem mond semmit arról, hogy mi legyen az összeadandókkal, a G pedig kiköti, hogy maradjanak változatlanok.

Felvetődik, hogy nem lenne elég a két komponensű állapottér is? Legyen $A = \mathbb{N} \times \mathbb{N}$ és

$$H = \{((a, b), (x, y)) \in A \times A \mid a + b = x\}.$$

Ezt a feladatot azonban nem úgy interpretálnánk, hogy "adjunk össze két természetes számot", hanem úgy, hogynöveljünknk meg egy természetes számot egy természetes számmal".

Megjegyezzük, gyakran fogalmaznak meg feladatot úgynevezett "input-output" modellben is, azaz milyen bemenő adatokhoz milyen kimenő adatokat rendelünk. Ez a szétválasztás az F és a G feladat esetében nem okozna gondot, de a H-hoz hasonló feladatok esetében már problémás lehetne, nem is beszélve a bevezetőben említett autós feladatról. Az állapotér modell igazi előnyeit a későbbiekben még tapasztalni fogjuk.

Felhívjuk a figyelmet arra, hogy a definíció szerint a feladat reláció, azaz általában nem determinisztikus, például G és H. A nem determinisztikusság azonban még "érdemibb" is lehet. Legyen a feladat a következő: határozzuk meg egy természetes szám egy valódi osztóját(a szám megváltoztatása nélkül)! Ebben az esetben $A = \mathbb{N} \times \mathbb{N}$ és

$$F = \{((a,b),(x,y)) \in A \times A \mid a = x \text{ és } y|x \text{ és } x \neq y \text{ és } y \neq 1\}.$$

Például $(6,5)$ pont F szerinti képe $\{(6,2),(6,3)\}$. A 6-nak 2 is, meg 3 is valódi osztója, azaz $|F(6,5)| = 2$.

Nagyon fontos, hogy pontosan lássuk a különbséget az előző feladat és a következő között: határozzuk meg egy természetes szám összes valódosztóját! Ebben az esetben az állapottér is más lesz, hiszen egy természetes szám összes valódosztójaja nem egy szám, hanem egy halmaz. Tehát $A' = \mathbb{N} \times \mathfrak{F}$, ahol $\mathfrak{F}$ az $\mathbb{N}$ véges részhalmazainak halmaza.

$$G = \{((a,b),(x,y)) \in A' \times A' \mid a = x \text{ és } y = \{n \in \mathbb{N} \mid n|x \text{ és } x \neq n \text{ és } n \neq 1\}\}.$$

Most $|G(6,\{5\})| = 1$ és $G(6,\{5\}) = \{(6,\{2,3\})\}$.

Megjegyezzük még, hogy $\mathcal{D}_F \neq A$, például $(5,5) \notin \mathcal{D}_F$, de $\mathcal{D}_G = A'$, például $(5,\{5\}) \in \mathcal{D}_G$ és $G(5,\{5\}) = \{(5,\{\})\}$.

2.3. A program

Amikor a program fogalmát igyekszünk tisztázni, a számítógépen futó programokra és az általuk megvalósított algoritmusokra figyelünk. Ha egy számítógépen egy program "fut", az abban jelentkezik, hogy a számítógép memóriájának tartalma folyamatosan változik. Itt most a "memóriát" általánosan értelmezzük, beleértünk a szűken vett memóriától, a regisztereken keresztül, a képernyőig mindent, ami információt hordoz.

A program jellemző tulajdonsága tehát, hogy "működik", azaz egy időben dinamikus folyamat. A dinamikus rendszerek általában nehezebbekezelhetőkök, vizsgálhatók, mint a statikusak. Ezért arra gondolunk, lehet-e helyettesíteni egy dinamikus folyamatot egy statikus modellel?

Tekintsük például az alábbi – a programozástól igazán messze eső – problémát: Adott egy kémiai kísérlet, amely túl gyorsan játszódik le ahhoz, hogy az ember pontosan regisztrálni tudja az egymásutáni eseményeket. Ez a programfutáshoz hasonlóan egy időben dinamikusan lejátszódó folyamat. Hogyan követhető nyomon mégis a kísérlet? Például úgy, hogy a kísérletet filmre vesszük, és a továbbiakban a képkockák által rögzített statikus állapotokat vizsgáljuk. Így az időben változó folyamatot egy statikus állapotsorozattal írjuk le.

A fenti példa szemléletesen mutatja, hogyan adhatunk statikus modellt egy dinamikus folyamat leírására.

A program definíciójában a program időbeni futásának jellemzésére az előbbpéldával példával analóg módon vezetünk be egy statikus modellt: a futást állapottérbeli sorozatokkal írjuk le. Ahogy a program futása során a memóriatartalom változik, úgy

jutunk az állapottér újabb és újabb pontjaiba, így ezeket a pontokat egy sorozatba fűzve valójában "fi lmre vesszük" a programfutást.

2.3. Definíció (PROGRAM). Programnak *nevezzük az* $S \subseteq A \times A^{**}$ *relációt, ha*

1. $\mathcal{D}_S = A$,
2. $\forall \alpha \in \mathcal{R}_S : \alpha = red(\alpha)$.
3. $\forall a \in A : \forall \alpha \in S(a) : \alpha_1 = a$,

A fenti defi nícióval a "működés" fogalmát akarjuk absztrakt módon megfogalmazni, ez magyarázza a sorozatokra vonatkozó kikötéseket. Az első tulajdonság azt jelenti, hogy a program az állapottér minden pontjához hozzárendel legalább egy sorozatot, azaz a program minden pontban "csinál" valamit. A "rosszul működést" is a működéssel, azaz valamilyen tulajdonságú sorozattal, sorozatokkal írjuk le.

A második tulajdonság azt fejezi ki, hogy a program értékkészlete olyan sorozatokból áll, amikben nem szerepel egymás után ugyanaz az elem. Egész pontosan, ha ez mégis előfordul, akkor ez az elem ismétlődik végtelen sokszor. A működés abban nyílvánul meg, hogy megváltozik az állapot, vagy ha mégsem az az abnormális működés jele. Emlékeztetünk arra, hogy az állapottér komponensei között minden előfordul előfodul, tehát ha bármi történiks töténik, az más állapotérbeli pontot jelent. A sorozatok között lehetnek "normális" sorozatok sorzatok is. Az, hogy az állapottér egy pontjához a program végtelen sorozatot rendel, azt jelenti, hogy a program futása nem fejeződik be.

A harmadik tulajdonság csak annyit jelent, hogy a sorozat a működés teljes történetét leírja, beleértve a kiinduló állapotot is, ezért azt, hogy egy program egy pontból elindítva nem csinál semmit, egy ebből az egy pontból álló, egy hosszúságú sorozat jellemzi.

A programot is relációként defi niáltuk, vagyis egy-egy állapottérbeli ponthoz több sorozat is hozzá lehet rendelve, azaz a működés nem determinisztikus. Ez első pillantásra talán meglepő, valójában természetes. Természetes akkor is ha számítógépen, számítógép rendszeren futó programra gondolunk, hiszen egy program sok processzorból, sok, akár egymástól nagy távolságban levő, komponensből álló rendszeren fut. De természetes akkor is, ha fi gyelembe vesszük, hogy a program fogalom nem csak a gépen futó program, hanem az algoritmus absztrakciója is, amik között lehetnek "nyitva" hagyott részek is.

2.4. A programfüggvény

Most visszatérünk a fejezet elején szereplő autós példához. A feladat az volt, hogy készítsük fel az autót egy hosszabb útra. Attól függően, hogy az autó milyen állapotban van, azaz a jellemzői milyen értékekkel rendelkeznek: mennyi benne amekkoran, mekora a nyomás a kerműködik, működik-e az irányjeltevékenységekékenségek sorozatát hajtjuk végre, felpumpáljuk a kerekeket, kicserélünk egy izzót és így tovább, lépésről lépésre változik az állapot, működik a program. Ha végül olyan állapotba jut az autó, hogy most már nyugodtan el lehet vele indulni egy hosszabb útra, akkor a program megoldotta a feladatot

Ahhoz, hogy egy program és egy feladat viszonyát megvizsgáljuk, elegendő, ha a programról tudjuk, hogy az állapottér egy adott pontjából kiindulva, az állapottér mely pontjába jut, mert a megoldás szempontjából a közbülső állapotok lényegtelenek. Természetesen vannak olyan – a programok minőségére vonatkozó – további kritériumok, amelyek szempontjából egyáltalán nem mindegy, hogy a program hogyan

oldja meg a feladatot (ilyen lehet például a hatékonyság, a program idő- és tárigénye), de a továbbiakban ezekkel egyelőre nem foglalkozunk.

Ezért vezetjük be a programfüggvény fogalmát, amely a program futásának eredményét jellemzi.

2.4. Definíció (PROGRAMFÜGGVÉNY). *A $p(S) \subseteq A \times A$ reláció az $S \subseteq A \times A^{**}$ program* programfüggvénye, *ha*

1. $\mathcal{D}_{p(S)} = \{a \in A \mid S(a) \subseteq A^*\}$,
2. $p(S)(a) = \{b \in A \mid \exists \alpha \in S(a) : \tau(\alpha) = b\}$.

Az első követelmény azt fogalmazza meg, hogy csak azokban a pontokban van értelme azt vizsgálni, hogy hova jut egy program, ahonnét kiindulva a program nem "száll el". A második pont értelemszerűen azt írja le, hogy ahova a program eljut, az a sorozat utolsó eleme.

Ha két program programfüggvénye megegyezik, az azt jelenti, hogy a két program működésének eredménye ugyanaz. Ezért mondjuk ebben az esetben azt, hogy a két program ekvivalens.

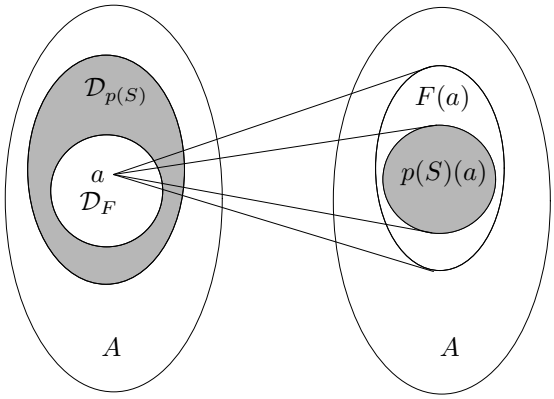
A programfüggvény elnevezés megtévesztő lehet, hiszen egy program programfüggvénye nem feltétlenül függvény, sőt az sem biztos, hogy determinisztikus reláció (parciális függvény). Jobban kifejezi a fogalom tartalmát a *hatásreláció* elnevezés. Mindkettőt használni fogjuk.

2.5. Megoldás

Fontos, hogy a programfüggvény ugyanolyan típusú reláció mint a feladat volt. Így tehát a programfüggvény fogalmának bevezetésével lehetőségünk nyílik arra, hogy kapcsolatot teremtsünk egy adott feladat és egy adott program között. Természetesen ennek a kapcsolatnak azt kell leírnia, hogy mikor mondjuk egy programról azt, hogy megold egy adott feladatot.

2.5. Definíció (MEGOLDÁS). *Azt mondjuk, hogy az S program* megoldja *az F feladatot, ha*

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



2.1. ábra. Megoldás

Ezzel a definícióval végül is azt kívánjuk meg, hogy az állapottér olyan pontjaihoz, ahol a feladat értelmezve van, a program csak véges sorozatokat rendeljen (terminál-jon) és a sorozatok végpontjait a feladat hozzárendelje a kezdőponthoz.

Néha gondot okoz ennek a definíciónak a megértése. Miért a programfüggvény szerinti kép része a feladat szerinti képnek? "Így nem kapunk meg minden megol-dást!" Pedig elég csak az autós példára gondolni. Például a fékfolyadék szintjének a minimum és a maximum szint jelzés között kell lenni. Ezt a karbantartás eredmény-eképpen teljesítjük, de egyáltalán nem egyételmű, hogy mi lesz a beállított szint. Annak meg nincs is értelme, hogy "minden lehetséges szintet" beállítsunk.

Sokszor felmerül a kérdés, hogy van-e összefüggés két feladat között a megoldás szempontjából? Ezzel kapcsolatos a következő definíció.

2.6. Definíció (SZIGORÍTÁS). *Azt mondjuk, hogy az $F_1 \subseteq A \times A$ feladat* szigorúbb *mint az $F_2 \subseteq A \times A$ feladat, ha*

1. $\mathcal{D}_{F_2} \subseteq \mathcal{D}_{F_1}$,
2. $\forall a \in \mathcal{D}_{F_2} : F_1(a) \subseteq F_2(a)$.

A szigorítás definíciójával összevetve a megoldás definíciójával könnyen adódik a következő egyszerű, de fontos tétel:

2.1. állítás: *Ha F_1 szigorúbb mint F_2 és S megoldása F_1-nek, akkor S megoldása F_2-nek is.*

2.6. Programozási feladat

Létezik-e tetszőleges feladathoz megoldó program? Legyen $F = A \times A$. Definiáljuk S-et a következőképpen: $\forall a \in \mathcal{D}_F : S(a) = \{red(\langle a, b\rangle) \mid b \in F(a)\}$ és $\forall a \notin \mathcal{D}_F : S(a) = \{\langle a\rangle\}$. Nyílvánvaló, hogy $p(S) = F$, tehát S megoldása F-nek. Vagyis tetszőleges feladathoz könnyen tudunk megoldó programot csinálni. Ebből kiindulva azt gondolhatnánk, hogy a programozás nagyon egyszerű feladat, ami persze nem igaz. A programozás feladata azonban ennél összetettebb. Egy programot adott programokból, rögzitett szabályok szerint kell összeraknunk, azaz egy programnyelv eszközeit kell használnunk.

2.7. Definíció (PROGRAMOZÁSI FELADAT). *Legyen $A = \overset{\times}{\underset{i \in I}{}} A_i$. Programozási feladatnak nevezzük az $(F, \mathbb{P}, \mathbb{K})$ hármast, ahol $F \subseteq A \times A$ egy feladat; $\mathbb{P}$ a megengedett programok halmaza, $\forall S \in \mathbb{P} : S \subseteq A \times A^{**}$; $\mathbb{K}$ a megengedett programkonstrukciók halmaza, $\forall K \in \mathbb{K}$ egy vagy több A-n értelmezett programhoz rendel egy A-n értelmezett programot.*

2.8. Definíció (PROGRAMOZÁSI FELADAT MEGOLDÁSA). *Az $(F, \mathbb{P}, \mathbb{K})$ programozási feladatnak az S program megoldása, ha S a megengedett programokból a megengedett konstrukciókkal előállítható és megoldása F-nek.*

A programozási feladat két értelemben is általánosítható: egyrészt kibővíthető a program működésére vonatkozó feltételekkel, ezzel a könyv második felében foglalkozunk , másrészt nem követeljük meg az azonos állapotteret, ehhez általánosítjuk a megoldás fogalmát, illetve bevezetjük a típusspecifikáció, típus, megfelelés és a típus-konstrukciók fogalmát.

Az, hogy milyen programkonstrukciókat engedünk meg, sokmindentől függ, mi a következőkben csak a legegyszerűbbekkel fogunk foglalkozni, mivel ezek is elégségesek egy programozási feladat megoldásához.

Már most felhívjuk a figyelmet arra a fontos szempontra, hogy valójában nagyon gyakran nem azt az utat követjük, hogy meglevő programokból rakjuk össze a megoldó programot, hanem a feladatot bontjuk fel rész feladatokra, úgy, hogy az ezeket megoldó programokból "automatikusan" megkapjuk az eredeti feladatot megoldó programot.

2.7. Példák

2.1. példa: Legyen $A_1 = \{1,2\},\ A_2 = \{1,2\},\ A_3 = \{1,2,3,4,5\},\ A = A_1 \times A_2 \times A_3$. $F = \{((a,b,c),(d,e,f)) \mid f = a+b\}$. $F(1,1,1) = ?$ Hány olyan pontja van az állapottérnek, amelyekhez a feladat ugyanazt rendeli, mint az $(1,1,1)$-hez?
Megoldás:
$F(1,1,1) = \{(1,1,2),(1,2,2),(2,1,2),(2,2,2)\}$.
Mivel a feladat hozzárendelése nem függ az állapottér harmadik komponensétől, a feladat ugyanezeket a pontokat rendeli az összes $(1,1,*)$ alakú ponthoz. Más pontokhoz viszont nem rendelheti ugyanezeket a pontokat, mert akkor az összeg nem lehetne 2! Tehát öt olyan pontja van az állapottérnek amelyhez a feladat ugyanazt rendeli, mint az $(1,1,1)$-hez.

2.2. példa: Legyen $A = \{1,2,3,4,5\}, S \subseteq A \times A^{**}$.

$$\begin{aligned} S = \{\ & (1,\langle 1251\rangle), && (1,\langle 14352\rangle), && (1,\langle 132\ldots\rangle), && (2,\langle 21\rangle), \\ & (2,\langle 24\rangle), && (3,\langle 333333\ldots\rangle), && (4,\langle 41514\rangle), && (4,\langle 431251\rangle), \\ & (4,\langle 41542\rangle), && (5,\langle 524\rangle), && (5,\langle 534\rangle), && (5,\langle 5234\rangle) && \} \end{aligned}$$

$F = \{(2,1)\,(2,4)\,(4,1)\,(4,2)\,(4,5)\}$.

a) Adjuk meg $p(S)$-t!

b) Megoldja-e S a feladatot?

Megoldás:

a) Mivel a program az 1-hez és a 3-hoz végtelen sorozatot is rendel, a programfüggvény értelmezési tartománya:

$$\mathcal{D}_{p(S)} = \{2,4,5\}.$$

Ekkor a programfüggvény:

$$p(S) = \{(2,1),(2,4),(4,4),(4,1),(4,2),(5,4)\}.$$

b) A megoldás definíciója két pontjának teljesülését kell belátnunk.

i. $\mathcal{D}_F = \{2,4\} \subseteq \{2,4,5\} = \mathcal{D}_{p(S)}$.

ii. $p(S)(2) = \{1,4\} \subseteq \{1,4\} = F(2)$,
$p(S)(4) = \{4,1,2\} \not\subseteq \{1,2,5\} = F(4)$,

tehát az S program nem megoldása az F feladatnak.

2.3. példa: Fejezzük ki a programok uniójának programfüggvényét a programok programfüggvényeivel!

Megoldás: Legyenek $S_1, S_2 \subseteq A \times A^{**}$ programok. Ekkor a programfüggvény értelmezési tartományának definíciójából kiindulva:

$$\begin{aligned}\mathcal{D}_{p(S_1 \cup S_2)} &= \{a \in A \mid p(S_1 \cup S_2)(a) \subseteq A^*\} = \\ &= \{a \in A \mid p(S_1)(a) \subseteq A^* \wedge p(S_2)(a) \subseteq A^*\} = \\ &= \mathcal{D}_{p(S_1)} \cap \mathcal{D}_{p(S_2)}.\end{aligned}$$

Legyen $a \in \mathcal{D}_{p(S_1)} \cap \mathcal{D}_{p(S_2)}$. Ekkor

$$\begin{aligned}p(S_1 \cup S_2)(a) &= \{\tau(\alpha) \mid \alpha \in (S_1 \cup S_2)(a)\} = \\ &= \{\tau(\alpha) \mid \alpha \in S_1(a) \vee \alpha \in S_2(a)\} = \\ &= p(S_1)(a) \cup p(S_2)(a).\end{aligned}$$

2.4. példa: Legyen F_1 és F_2 egy-egy feladat ugyanazon az állapottéren! Igaz-e, hogy ha minden program, ami megoldása F_1-nek, az megoldása F_2-nek is, és minden program, ami megoldása F_2-nek, az megoldása F_1-nek is, akkor F_1 és F_2 megegyeznek?
Megoldás: A leggyakoribb hiba, amit ennek a feladatnak a megoldásakor el szoktak követni, az az, hogy összekeverik az állítás feltételrendszerét magával a bizonyítandó állítással, és azt próbálják bebizonyítani, hogy valamelyik feladatnak minden program megoldása. Természetesen általában ez nem igaz, de nem is ez a feladat! Abból kell tehát kiindulnunk, hogy pontosan ugyanazok a programok oldják meg mindkét feladatot, és meg kell vizsgálnunk, hogy következik-e ebből az, hogy a két feladat megegyezik.

Induljunk ki abból, hogy minden program, ami megoldása F_1-nek, az megoldása F_2-nek, és válasszunk egy olyan programot, amelynek programfüggvénye megegyezik az F_1 relációval. Ekkor a választott program triviálisan megoldja az F_1 feladatot, tehát meg kell oldania F_2-t is, azaz:

$$\begin{aligned}&i. \quad \mathcal{D}_{F_2} \subseteq \mathcal{D}_{F_1}, \\ &ii. \quad \forall a \in \mathcal{D}_{F_2} : F_1(a) \subseteq F_2(a)\end{aligned}$$

Most felhasználva, hogy minden program, ami megoldása F_2-nek, az megoldása F_1-nek is, és egy olyan program választásával, amelynek programfüggvénye megegyezik F_2-vel, az előzőekkel analóg módon adódnak a fordított irányú állítások:

$$\begin{aligned}&iii. \quad \mathcal{D}_{F_1} \subseteq \mathcal{D}_{F_2}, \\ &iv. \quad \forall a \in \mathcal{D}_{F_1} : F_2(a) \subseteq F_1(a).\end{aligned}$$

Az $i.$ és $iii.$ állításokból következik, hogy a két feladat értelmezési tartománya megegyezik, míg az $ii.$ és $iv.$ állítások garantálják, hogy ezen közös értelmezési tartomány egyes pontjaihoz mindkét feladat ugyanazokat a pontokat rendeli, azaz $F_1 = F_2$.

2.5. példa: $F_1 \subseteq F_2$. Az S program megoldja F_2-t. Igaz-e, hogy S megoldja F_1-et is?
Megoldás: Próbáljuk meg bebizonyítani az állítást. Ehhez a megoldás definíciója két pontját kell belátnunk.

$i.$ $\mathcal{D}_{F_1} \subseteq \mathcal{D}_{p(S)}$,

$ii.$ $\forall a \in \mathcal{D}_{F_1} : p(S)(a) \subseteq F_1(a)$.

Az $i.$ pont teljesülése könnyen látható, ugyanis S megoldása F_2-nek, tehát

$$\mathcal{D}_{F_1} \subseteq \mathcal{D}_{F_2} \subseteq \mathcal{D}_{p(S)}.$$

Az $ii.$ pont bizonyításánál azonban gond van, hiszen az alábbi két állítás áll rendelkezésünkre:

$$\forall a \in \mathcal{D}_{F_1}: \quad p(S)(a) \subseteq F_2(a),$$
$$\forall a \in \mathcal{D}_{F_1}: \quad F_1(a) \subseteq F_2(a).$$

és ezekből a kívánt állítás nem bizonyítható. Elakadtunk a bizonyításban, lehet, hogy nem igaz az állítás? Készítsünk ellenpéldát felhasználva azt, hogy hol akadtunk el a bizonyításban!

Legyen $A = \{1, 2\}$, $F_1 = \{(1, 1)\}$, $F_2 = \{(1, 1), (1, 2)\}$ és $p(S)$ egyezzen meg az F_2 feladattal. Ekkor S triviálisan megoldja F_2-t, de nem megoldása F_1-nek, ugyanis

$$1 \in \mathcal{D}_{F_1} \wedge p(S)(1) = F_2(1) = \{1, 2\} \not\subseteq \{1\} = F_1(1).$$

Tehát az állítás nem igaz.

2.6. példa: Legyenek S_1 és $S_2 \subseteq A \times A^{**}$ programok, $F \subseteq A \times A$ pedig Tegyükadat. Teggyük fel továbbá, hogy $S_1 \subseteq S_2$ és S_2 megoldja az F feladatot. Igaz-e, hogy S_1 megoldja F-et?
Megoldás: Ha $S_1 \subseteq S_2$, akkor mit tudunk a programfüggvényüelőszörNézzük előszö az értelmezési tartományokat! A defi níció szerint minden állapottérbeli ponthoz minden program hozzárendel legalább egy sorozatot, így S_1 és S_2 is. Mivel $S_1 \subseteq S_2$ ezért csak az fordulhat elő, hogy egy adott ponthoz S_2 olyan sorozatokat is rendel, amit S_1 nem. Ha ezek a sorozatok mind végesek, akkor az adott pont vagy benne van mindkét program programfüggvényének az értelmezési tartományában, vagy egyikében sem, ha van közöttük végtelen is, az adott pont biztosan nem eleme $p(S_2)$ értelmezési tartományának, de eleme lehet $\mathcal{D}p(S_1)$-nek. Tehát $\mathcal{D}p(S_2) \subseteq \mathcal{D}p(S_1)$ és $\forall a \in \mathcal{D}p(S_2) : p(S_1)(a) \subseteq p(S_2)(a)$.

Mivel S_2 megoldása F-nek, ezért $\mathcal{D}F \subseteq \mathcal{D}p(S_2)$ és $\forall a \in \mathcal{D}F : p(S_2)(a) \subseteq F(a)$. A fentiek miatt $\mathcal{D}F \subseteq \mathcal{D}p(S_1)$ is és $\forall a \in \mathcal{D}F : p(S_1)(a) \subseteq F(a)$ is teljesül, vagyis S_1 is megoldása F-nek.

2.8. Feladatok

2.1. Legyen $A = \{\Omega, \Phi, \Psi, \Theta, \Gamma\}, S \subseteq A \times A^{**}$.

$$\begin{aligned} S = \{ \quad & (\Omega, \langle \Omega\Phi\Gamma\Omega \rangle), && (\Omega, \langle \Omega\Theta\Psi\Gamma \rangle), && (\Omega, \langle \Omega\Psi\Phi \ldots \rangle), \\ & (\Phi, \langle \Phi\Omega \rangle), && (\Psi, \langle \Psi\Theta \rangle), && (\Psi, \langle \Psi\Psi\Psi\Psi\Psi\Psi \ldots \rangle), \\ & (\Theta, \langle \Theta\Omega\Gamma\Omega\Theta \rangle), && (\Theta, \langle \Theta\Psi\Omega\Phi\Gamma\Omega \rangle), && (\Theta, \langle \Theta\Omega\Gamma\Theta\Phi \rangle), \\ & (\Gamma, \langle \Gamma\Phi\Psi \rangle), && (\Gamma, \langle \Gamma\Psi \rangle), && (\Gamma, \langle \Gamma\Phi\Psi\Omega \rangle) \qquad \} \end{aligned}$$

$$F = \{(\Phi, \Omega)\,(\Phi, \Psi)\,(\Theta, \Omega)\,(\Theta, \Phi)\,(\Theta, \Theta)\}.$$

a) Adjuk meg $p(S)$-t!

b) Megoldja-e S a feladatot?

2.2. Legyen S program, F olyan feladat, hogy S megoldása F-nek. Igaz-e, hogy

a) ha F nem determinisztikus, akkor S sem az?

b) ha F determinisztikus, akkor S is az?

c) ha F nem determinisztikus, akkor $p(S)$ sem az?

d) ha $p(S)$ determinisztikus, akkor F is az?

e) ha F determinisztikus, akkor $p(S)$ is az?

f) ha S nem determinisztikus, akkor $p(S)$ sem az?

2.3. Igaz-e, hogy $p(S)$ értelmezési tartománya éppen A^* ősképe S-re nézve?

2.4. Mondhatjuk-e, hogy az S program megoldja az F feladatot, ha igaz a következő állítás:

$$q \in \mathcal{D}_F \Rightarrow S(q) \subseteq A^* \wedge p(S)(q) \subseteq F(q).$$

2.5. Legyen $A = \mathbb{N} \times \mathbb{N}$, $F_1, F_2 \subseteq A \times A$.

$F_1 = \{((u,v),(x,y)) \mid \ \ y|u\}$,

$F_2 = \{((u,v),(x,y)) \mid x = u \wedge y|u\}$.

Ugyanaz-e a két feladat? (Van-e valamilyen összefüggés közöttük?)

2.6. $F \subseteq A \times A$. S_1, S_2 programok A-n. Az S_1 és az S_2 is megoldja az F feladatot. Igaz-e, hogy az $S = (S_1 \cup S_2)$ program is megoldja az F feladatot?

2.7. Tekintsük a következő szövegesen megadott feladatot: Adott egy sakktábla, és két rajta lévő bástya helyzete. Helyezzünk el a táblán egy harmadik bástyát úgy, hogy az mindkettőnek az ütésében álljon! Készítsük el a modellt: írjuk fel az állapotteret és az F relációt!

2.8. Tudjuk, hogy S megoldja F-et (az A állapottéren). Igaz-e, hogy

$$(a \in A \wedge (S(a) \not\subseteq A^* \vee p(S)(a) \not\subseteq F(a))) \Rightarrow a \notin \mathcal{D}_F?$$

2.9. Legyen $F \subseteq A \times A$ egy feladat és $S \subseteq A \times A^{**}$ egy program. Jelöljük FP-vel azt a relációt, amely F és $p(S)$ metszeteként áll elő. Igaz-e, hogy

a) ha $\mathcal{D}_{FP} = \mathcal{D}_F$, akkor S megoldja F-et?

b) ha S megoldja F-et, akkor $\mathcal{D}_{FP} = \mathcal{D}_F$?

3. fejezet

Specifikáció

A megoldás definíciója közvetlenül elég nehézkesen használható a programok készítése során, hiszen az, hogy egy program megold-e egy feladatot az a megoldás eddigi definíciója alapján csak nehezen ellenőrizhető. Ezért bevezetünk néhány új fogalmat, majd ezek segítségével megadjuk a megoldás egy elégséges feltételét.

3.1. A leggyengébb előfeltétel

Először a program működésének eredményét adjuk meg egy a programfüggvénynél kényelmesebben használható jellemzővel.

3.1. Definíció (LEGGYENGÉBB ELŐFELTÉTEL). *Legyen $S \subseteq A \times A^{**}$ program, $R : A \to \mathbb{L}$ állítás. Ekkor az S program R utófeltételhez tartozó* leggyengébb előfeltétele *az a $lf(S,R) : A \to \mathbb{L}$, függvény amelyre:*

$$\lceil lf(S,R) \rceil = \{a \in \mathcal{D}_{p(S)} | p(s)(a) \subseteq \lceil R \rceil\}.$$

A leggyengébb előfeltétel tehát pontosan azokban a pontokban igaz, ahonnét kiindulva az S program biztosan terminál, és az összes lehetséges végállapotra igaz R.

Természetesen a leggyengébb előfeltétel igazsághalmazán kívül is lehetnek olyan pontok, amelyből a program egy futása eljut az utófeltétel igazsághalmazába, csak azokból a pontokból nem garantált, hogy oda jut.

Egy program működése úgy is jellemezhető, hogy megadjuk a program tetszőleges utófeltételhez tartozó leggyengébb előfeltételét. A feladat megoldása során az a célunk, hogy olyan programot találjunk, amelyik bizonyos feltételeknek eleget tevő pontokban terminál. Ezért azt mondhatjuk, hogy ha a számunkra kedvező végállapotokra megadjuk a program leggyengébb előfeltételét, akkor a programfüggvény meghatározása nélkül jellemezzük a program működését.

Emlékeztetünk arra, definiáltuk a reláció szerinti őskép fogalmát is, ennek felhasználásával azonnal látszik, hogy

$$\lceil lf(S,R) \rceil = p(S)^{-1}(\lceil R \rceil).$$

Felhasználva az igazsághalmaz definícióját és a szigorú kompozíció szerinti őskép tulajdonságát

$$p(S)^{-1}(\lceil R \rceil) = p(S)^{-1}(R^{-1}(\{igaz\})) = (R \odot p(S))^{-1}(\{igaz\}) = \lceil R \odot p(S) \rceil.$$

Mivel R függvény, a kompozíció és a szigorú kompozíció megegyezik, tehát

$$\lceil lf(S,R)\rceil = \lceil R \circ p(S)\rceil.$$

Abban az esetben, ha $p(S)$ is függvény $lf(S,R) = R \circ p(S)$.

A fenti összefüggésekre gyakran fogunk hivatkozni.

A most következő tétel a leggyengébb előfeltétel néhány nevezetes tulajdonságáról szól.

3.1. TÉTEL: A lf TULAJDONSÁGAI

Legyen $S \subseteq A \times A^{**}$ program, $Q, R : A \to \mathbb{L}$ állítások. Ekkor

(1) $lf(S, HAMIS) = HAMIS$,

(2) Ha $Q \Rightarrow R$, akkor $lf(S,Q) \Rightarrow lf(S,R)$,

(3) $lf(S,Q) \wedge lf(S,R) = lf(S, Q \wedge R)$,

(4) $lf(S,Q) \vee lf(S,R) \Rightarrow lf(S, Q \vee R)$.

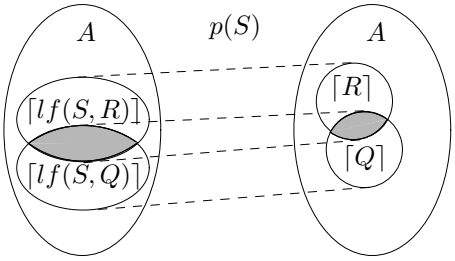
Az első tulajdonságot a csoda kizárása elvének, a másodikat monotonitási tulajdonságnak nevezzük.

Bizonyítás:

1. Indirekt: Tegyük fel, hogy $\exists a \in \lceil lf(S, HAMIS)\rceil$. Ekkor a leggyengébb előfeltétel definíciója szerint: $a \in \mathcal{D}_{p(S)}$ és $p(S)(a) \subseteq \lceil HAMIS\rceil = \emptyset$. Ez nyilvánvaló ellentmondás.

2. Indirekt: Tegyük fel, hogy $\exists a \in \lceil lf(S,Q)\rceil \setminus \lceil lf(S,R)\rceil$. Ekkor $a \in \mathcal{D}_{p(S)}$ és $p(S)(a) \subseteq \lceil Q\rceil \wedge p(S)(a) \not\subseteq \lceil R\rceil$. Ez viszont ellentmond annak a feltételnek, mely szerint $\lceil Q\rceil \subseteq \lceil R\rceil$

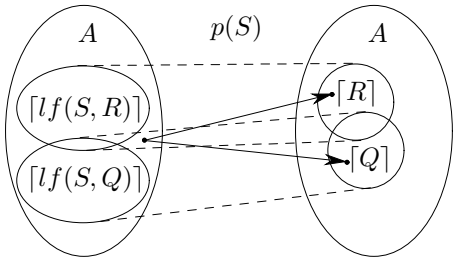
3. Az állítást két részben, a mindkét irányú következés belátásával bizonyítjuk.



3.1. ábra. A leggyengébb előfeltétel és a metszet kapcsolata

(a) $lf(S,Q) \wedge lf(S,R) \Rightarrow lf(S, Q \wedge R)$, ugyanis:
Legyen $a \in \lceil lf(S,Q) \wedge lf(S,R)\rceil$. Ekkor $a \in \lceil lf(S,Q)\rceil$ és $a \in \lceil lf(S,R)\rceil$, azaz $a \in \mathcal{D}_{p(S)}$ és $p(S)(a) \subseteq \lceil Q\rceil$, illetve $p(S)(a) \subseteq \lceil R\rceil$. Ekkor azonban $p(S)(a) \subseteq \lceil Q\rceil \cap \lceil R\rceil = \lceil Q \wedge R\rceil$, azaz $a \in \lceil lf(S, Q \wedge R)\rceil$.

(b) $lf(S, Q \wedge R) \Rightarrow lf(S,Q) \wedge lf(S,R)$, ui.:
Legyen $a \in \lceil lf(S, Q \wedge R)\rceil$. Ekkor a leggyengébb előfeltétel definíciója alapján $a \in \mathcal{D}_{p(S)}$ és $p(S)(a) \subseteq \lceil Q \wedge R\rceil$. Felhasználva, hogy $\lceil Q \wedge R\rceil = \lceil Q\rceil \cap \lceil R\rceil$, adódik, hogy $p(S)(a) \subseteq \lceil Q\rceil$ és $p(S)(a) \subseteq \lceil R\rceil$, azaz $a \in \lceil lf(S,Q)\rceil$ és $a \in \lceil lf(S,R)\rceil$, tehát $a \in \lceil lf(S,Q) \wedge lf(S,R)\rceil$.



3.2. ábra. A leggyengébb előfeltétel és az unió kapcsolata

4. Legyen $a \in \lceil lf(S,Q) \vee lf(S,R) \rceil$. Ekkor $a \in \lceil lf(S,Q) \rceil$ vagy $a \in \lceil lf(S,R) \rceil$. Ha $a \in \lceil lf(S,Q) \rceil$, akkor – a monotonitási tulajdonság alapján – $a \in \lceil lf(S, Q \vee R) \rceil$. Hasonlóan ha $a \in \lceil lf(S,R) \rceil$, akkor $a \in \lceil lf(S, Q \vee R) \rceil$.

□

A tulajdonságoság visszafelé nem igaz. $p(S)(a) \subseteq \lceil Q \rceil \cup \lceil R \rceil$ nem következik sem $p(S)(a) \subseteq \lceil Q \rceil$, sem $p(S)(a) \subseteq \lceil R \rceil$. Természetesen, ha $p(S)$ determinisztikus, azaz $\forall a \in A : p(S)(a)$ legfeljebb egy elemű halmaz, akkor az egyenlőség fennáll.

3.2. A feladat specifikációja

A következőkben bevezetjük a feladat megadásának egy másik módját, és kimondunk egy a gyakorlat szempontjából nagyon fontos tételt.

Általában a feladat nem függ az állapottér összes komponensétől, azaz az állapottér több pontjához is ugyanazt rendeli. Ezeket a pontokat fogjuk össze egy ponttá a paramétertér segítségével.

3.2. Definíció (PARAMÉTERTÉR). *Legyen $F \subseteq A \times A$ feladat. A B halmazt a feladat* paraméterterének *nevezzük, ha van olyan F_1 és F_2 reláció, hogy*

$$\begin{aligned} F_1 &\subseteq A \times B, \\ F_2 &\subseteq B \times A, \\ F &= F_2 \circ F_1. \end{aligned}$$

Fontos észrevenni, hogy paraméterteret mindig lehet találni. Például maga a feladat állapottere minden esetben választható paramétertérnek úgy, hogy a definícióban szereplő F_1 relációnak az identikus leképezést, F_2-nek pedig magát az F feladatot választjuk. Ám az, hogy egy konkrét esetben mit is választunk paramétertérnek a feladattól függ. Általában úgy választjuk meg a paraméterteret, hogy a következő tételt kényelmesen tudjuk használni.

3.2. TÉTEL: SPECIFIKÁCIÓ TÉTELE

Legyen $F \subseteq A \times A$ feladat, B az F egy paramétertere, $F_1 \subseteq A \times B$, $F_2 \subseteq B \times A$, $F = F_2 \circ F_1$. Legyen $b \in B$, és definiáljuk a következő állításokat:

$$\begin{aligned} \lceil Q_b \rceil &= \{a \in A \mid (a,b) \in F_1\} = F_1^{(-1)}(b) \\ \lceil R_b \rceil &= \{a \in A \mid (b,a) \in F_2\} = F_2(b). \end{aligned}$$

Ekkor ha $\forall b \in B : Q_b \Rightarrow lf(S, R_b)$, akkor az S program megoldja az F feladatot.

Bizonyítás: A megoldás definíciója két pontjának teljesülését kell belátnunk:

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

   Legyen $a \in \mathcal{D}_F$ tetszőleges. Ekkor az F_1 és F_2 relációk definíciója miatt $a \in \mathcal{D}_{F_1}$ és

   $$\exists b \in B : a \in \lceil Q_b \rceil.$$

   De ekkor a tétel feltétele alapján:

   $$a \in \lceil Q_b \rceil \subseteq \lceil lf(S, R_b) \rceil \subseteq \mathcal{D}_{p(S)}.$$

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

   Legyen $a \in \mathcal{D}_F$ tetszőlegesen rögzített, $b \in B$ olyan, amelyre $a \in \lceil Q_b \rceil$. Ekkor a feltétel szerint:

   $$p(S)(a) \subseteq \lceil R_b \rceil = F_2(b) \subseteq F_2(F_1(a)) = F(a).$$

□

A specifikáció tétele csak elégséges feltétel a megoldásra, azaz nem megfordítható: lehet adni olyan feladat-program párt, ahol a program megoldja a feladatot, de a specifikáció tétele nem teljesül. Ez természetesen attól is függ, hogy a feladatot hogyan specifikáljuk, azaz milyen paraméterteret választunk, és hogyan bontjuk a feladatot F_1 és F_2 relációk kompozíciójára.

Azonnal látszik, hogy

$$\bigcup_{b \in B} \lceil Q_b \rceil = \mathcal{D}_{F_1} \supseteq \mathcal{D}_F.$$

Ha egy $b \in B$-re $\lceil Q_b \rceil \not\subseteq \mathcal{D}_F$, akkor $\lceil R_b \rceil = \emptyset$.

3.3. A változó fogalma

Az eddig elmondottakból alapján a specifikáció tétele még nem lenne hatékonyan használható, hiszen a paramétertér minden pontjára ellenőriznünk kellene a feltételek teljesülését. Ezért bevezetjük a változó fogalmát, aminek segítségével a feltételrendszer teljesülése egyszerűbben ellenőrizhetővé válik.

3.3. Definíció (VÁLTOZÓ). *Az* $A = \overset{\times}{\underset{i \in I}{}} A_i$ *állapottér* $v_i : A \to A_i$ *egydimenziós projekciós függvényeit változóknak nevezzük.*

A változók használatával egyszerűsíthetjük az állapottéren értelmezett állítások (elő- és utófeltételek, leggyengébb előfeltétel) és relációk (programfüggvény) leírását.

Mivel minden változó értelmezési tartománya az állapottér és értékkészlete egy típusértékhalmaz, egy változót jellemezhetünk egy típussal, azaz beszélhetünk a változó típusáról.

Ha a paramétertér is direktszorzat alakú – márpedig ez gyakran így van, ugyanis általában az állapottér egy altere – akkor a paramétertér egydimenziós projekciós függvényeit paraméterváltozóknak nevezzük.

Az állapottér illetve a paramétertér egyes komponenseihez a változókat, illetve paraméterváltozókat az adott komponens alá írjuk.

Most megvizsgáljuk, hogyan lehet a specifikáció tétele segítségével feladatokat megfogalmazni.

Tekintsünk egy már ismert feladatot: határozzuk meg két egész szám összegét!

Először felírjuk az állapotteret, úgy mint eddig, csak kiegészítjük a változó nevek megadásával.

$$\begin{array}{l} A = \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} \\ \quad\ x \quad y \quad z \end{array}$$

Az eddigi jelöléseink alkalmazásával a feladat

$$F = \{((u_1, u_2, u_3), (v_1, v_2, v_3)) \in A \times A \mid v_3 = u_1 + u_2\}.$$

A specifikáció tétele alkalmazásához írjuk föl a paraméterteret is:

$$\begin{array}{l} B = \mathbb{Z} \times \mathbb{Z} \\ \quad\ x' \quad y' \end{array}$$

Majd még visszatérünk arra, hogy miért éppen így választottuk meg a paraméter teret.

Tehát az állapottér három egész komponensből áll, melyeknek változói rendre x, y és z. A paramétertér két egész komponensből áll, az első komponens változója x', a másodiké y'.

Legyen az F_1 reláció a következő:

$$F_1 = \{((u_1, u_2, u_3), (b_1, b_2)) \in A \times B \mid u_1 = b_1 \text{ és } u_2 = b_2\},$$

F_2 pedig

$$F_2 = \{((b_1, b_2), (v_1, v_2, v_3)) \in B \times A \mid v_3 = b_1 + b_2\}.$$

A fentiekből adódik, hogy $F_2 \circ F_1 = F$. A paramétertér egy tetszőleges b eleméhez tartozó elő- és utófeltételre:

$$\begin{array}{rcl} \lceil Q_b \rceil & = & \{(a_1, a_2, a_3) \in A \mid a_1 = b_1 \text{ és } a_2 = b_2\} \\ \lceil R_b \rceil & = & \{(a_1, a_2, a_3) \in A \mid a_3 = b_1 + b_2\} \end{array}$$

amit az állapottér és a paramétertér változóinak felhasználásval is fölírhatunk:

$$\begin{array}{rcl} \lceil Q_b \rceil & = & \{a \in A \mid x(a) = x'(b) \text{ és } y(a) = y'(b)\} \\ \lceil R_b \rceil & = & \{a \in A \mid z(a) = x'(b) + y'(b)\}. \end{array}$$

A függvénytereknél tárgyaltuk, hogy a függvénytér elemein a relációk egy logikai függvényekből álló teret generálnak. x, y, z egy függvénytér elemei, $x'(b), y'(b)$ szintén annak tekinthetők, konstans függvények. Ezért $x = x'(b)$ és $y = y'(b)$ az állapottéren értelmezett logikai függvények, ahogy $x = x'(b) \wedge y = y'(b)$ is az. Ebből következik, hogy

$$\begin{array}{rcl} \lceil Q_b \rceil & = & \lceil x = x'(b) \wedge y = y'(b) \rceil \\ \lceil R_b \rceil & = & \lceil z = x'(b) + y'(b) \rceil \end{array}$$

és mivel minddegyik függvény,

$$\begin{array}{rcl} Q_b & = & (x = x'(b) \wedge y = y'(b)) \\ R_b & = & (z = x'(b) + y'(b)) \end{array}$$

A jelölés egyszerűsíthető, mivel nyílvánvaló, hogy a paraméter változók argumentuma b, ezek el is hagyhatók, sőt, általában az sem okoz félreértést, ha az elő- utófeltételek indexeit elhagyjuk. Ezek a feltételek a paramétertér pontjaihoz tartoznak és így a paraméterváltozók értékeitől függnek. A feladat specifikációja tehát:

$$
\begin{array}{lccc}
A = & \mathbb{Z} & \times \mathbb{Z} & \times \mathbb{Z} \\
 & x & y & z \\
B = & \mathbb{Z} & \times \mathbb{Z} & \\
 & x' & y' &
\end{array}
$$

$Q : (x = x' \wedge y = y')$

$R : (z = x' + y')$

A továbbiakban a feladatot úgy definiáljuk, hogy megadjuk az állapotterét (A), a paraméterterét (B), valamint az elő- és utófeltételét (Q illetve R) a paramétertér minden pontjára, azaz paraméteresen. Ebben az esetben azt mondjuk, hogy a feladatot megadtuk a specifikáció tételének megfelelő formában, vagy ha nem okoz félreértést, specifikáltuk a feladatot.

Egy feladatot nagyon sokféleképpen lehet specifikálni, a sok lehetőség közül az egyik az, amit elő-, utófeltétellel történő specifikációnak szoktak nevezni és nagyon hasonlít arra, amit most tárgyalunk. Felhívjuk a figyelmet, hogy a hasonlóság ellenére a kettő nem azonos.

Paramétertérnek általában az állapottér egy alterét szoktuk választani. Azokat a komponenseket válogatjuk ki, amelyek értékétől függ, hogy a feladat mihez, mit rendel, amik *paraméterezik* a feladatot. Lényegében azt fogalmazzuk meg, hogy az állapottér milyen tulajdonságú pontjaiból, milyen tulajdonságú pontokba akarunk jutni. A paraméterteret arra használjuk, hogy megadjuk milyen összefüggés van az elérendő és a kiinduló állapotok között.

Ha egy programnak meg tudjuk határozni a (paraméteres) utófeltételhez tartozó leggyengébb előfeltételét, akkor a specifikáció tétele alapján, könnyen eldönthetjük, hogy megoldása-e a specifikált feladatnak, más szóval *bebizonyíthatjuk a program helyességét*. Megjegyezzük azonban, hogy legtöbbször a "fordított" utat fogjuk követni, nem a program helyességét bizonyítjuk, hanem bizonyítottan helyes programot állítunk elő.

A későbbiekben bevezetünk majd olyan eszközöket, amelyek segítségével a feladat specifikációjából kiindulva olyan programokat készíthetünk, amelyek megoldják a feladatot.

3.4. Példák

3.1. példa: Legyen $A = \{Keats, Bach, Mozart, Liszt, Poe, Byron\}$, $S \subseteq A \times A^{**}$ program.

$$
\begin{array}{lllll}
S = \{ & Keats & \rightarrow \langle Keats, Bach\rangle, & Bach & \rightarrow \langle Bach, Mozart\rangle, \\
 & Bach & \rightarrow \langle Bach, Liszt, Byron\rangle, & Mozart & \rightarrow \langle Mozart, Keats\rangle, \\
 & Liszt & \rightarrow \langle Liszt, Byron\rangle, & Poe & \rightarrow \langle Poe, Mozart\rangle, \\
 & Byron & \rightarrow \langle Byron, Bach, Liszt\rangle\} & &
\end{array}
$$

Legyen továbbá az $R : A \rightarrow \mathbb{L}$ állítás:

$\forall x \in A : \ R(x) = (x$ zeneszerző).

Mi lesz a fenti program R-hez tartozó leggyengébb előfeltétele?

Megoldás: Írjuk fel először a program programfüggvényét:

$$
\begin{array}{lccc}
p(S) = \{ & (Keats, Bach), & (Bach, Mozart), & (Bach, Byron), \\
 & (Mozart, Keats), & (Liszt, Byron), & (Poe, Mozart), \\
 & (Byron, Liszt) & \} &
\end{array}
$$

Ezek után, a leggyengébb előfeltétel definíciót felhasználva:
$\lceil lf(S,R)\rceil = \{Keats, Poe, Byron\}$, ugyanis

$$\begin{aligned} p(S)(Keats) &= \{Bach\} \subseteq \lceil R\rceil \\ p(S)(Poe) &= \{Mozart\} \subseteq \lceil R\rceil \\ p(S)(Byron) &= \{Liszt\} \subseteq \lceil R\rceil \\ p(S)(Bach) &= \{Mozart, Byron\} \not\subseteq \lceil R\rceil \\ p(S)(Mozart) &= \{Keats\} \not\subseteq \lceil R\rceil \\ p(S)(Liszt) &= \{Byron\} \not\subseteq \lceil R\rceil \end{aligned}$$

3.2. példa: Legyen $H_1, H_2 : A \to \mathbb{L}$. Igaz-e, hogy ha minden $S \subseteq A \times A^{**}$ programra $lf(S,H_1) = lf(S,H_2)$, akkor $\lceil H_1\rceil = \lceil H_2\rceil$?
Megoldás: Felhasználva, hogy a leggyengébb előfeltételek minden programra megegyeznek, egy alkalmas program választásával a válasz egyszerűen megadható: rendelje az S program az állapottér minden eleméhez az önmagából álló egy hosszúságú sorozatot. Ekkor könnyen látható, hogy tetszőleges R utófeltétel esetén:

$$lf(S,R) = R.$$

Ekkor viszont

$$H_1 = lf(S,H_1) = lf(S,H_2) = H_2,$$

tehát a két feltétel megegyezik.

3.3. példa: Specifikáljuk a következő feladatot: $A = \mathbb{L} \times \mathbb{L}, \quad F \subseteq A \times A$,

$$F = \{((l,k),(l',k')) \mid k' = k \wedge l' = (l \wedge k)\}$$

Megoldás:

$$\begin{array}{l} A = \mathbb{L} \times \mathbb{L} \\ \quad\; x \quad y \\ B = \mathbb{L} \times \mathbb{L} \\ \quad\; x' \quad y' \end{array}$$

$Q : (x = x' \wedge y = y')$
$R : (x = (x' \wedge y') \wedge y = y')$

3.4. példa: Legyen $F \subseteq A \times A$, $S \subseteq A \times A^{**}$ program, B egy tetszőleges halmaz. Legyenek továbbá $F_1 \subseteq A \times B$ és $F_2 \subseteq B \times A$ olyan relációk, hogy $F = F_2 \circ F_1$, valamint $\forall b \in B$:

$$\begin{aligned} \lceil \widehat{Q}_b\rceil &= F_1^{-1}(b) \\ \lceil R_b\rceil &= F_2(b). \end{aligned}$$

Igaz-e, hogy ha $\forall b \in B : \widehat{Q}_b \Rightarrow lf(S,R_b)$, akkor S megoldja F-et?
Megoldás: Próbáljuk meg a megoldás definíciója két pontját belátni. Legyen $a \in \mathcal{D}_F$. Be kellene látnunk, hogy $a \in \mathcal{D}_{p(S)}$. Nézzük meg a specifikáció tételének bizonyítását: ott felhasználtuk, hogy ekkor van olyan $b \in B$, hogy $a \in \lceil Q_b\rceil$. Igaz ez a $\widehat{Q}_b$-re is? Sajnos – mivel $\widehat{Q}_b$-t ősképpel definiáltuk, ez nem feltétlenül van így. Próbáljunk a fenti gondolatmenet alapján ellenpéldát adni:

Legyen $A = \{1\}$, $B = \{1,2\}$, $F = \{(1,1)\}$, $F_1 = \{(1,1),(1,2)\}$, $F_2 = \{(2,1)\}$. Ekkor $\widehat{Q}_1 = hamis$ és $\widehat{Q}_2 = hamis$, tehát az állítás feltételei teljesülnek függetlenül a programtól (ui. „hamisból minden következik"). Válasszuk most az alábbi programot: $S = \{(1, <1,1,...>)\}$. Ez a program nem megoldása a feladatnak, de teljesülnek rá is az állítás feltételei. Tehát az állítás nem igaz.

3.5. Feladatok

3.1. Legyen $A = \{1,2,3,4,5\}, S \subseteq A \times A^{**}$.

$$\begin{array}{lllll} S = \{ & (1,\langle 1251\rangle), & (1,\langle 14352\rangle), & (1,\langle 132\ldots\rangle), & (2,\langle 21\rangle), \\ & (2,\langle 24\rangle), & (3,\langle 333333\ldots\rangle), & (4,\langle 41514\rangle), & (4,\langle 431251\rangle), \\ & (4,\langle 41542\rangle), & (5,\langle 524\rangle), & (5,\langle 534\rangle), & (5,\langle 5234\rangle) \quad \} \end{array}$$

és $\lceil R \rceil = \{1,2,5\}$. írd fel az $\lceil lf(S,R) \rceil$ halmazt!

3.2. Mivel egyenlő $lf(S, \text{IGAZ})$?

3.3. Legyen A tetszőleges állapottér, $Q_i : A \to \mathbb{L} \qquad (i \in \mathbb{N})$. Igaz-e, ha

$$\forall i \in \mathbb{N} : \; Q_i \Rightarrow Q_{i+1},$$

akkor

$$(\exists n \in \mathbb{N} : \; lf(S, Q_n)) = lf(S, (\exists n \in \mathbb{N} : \; Q_n))?$$

3.4. Igaz-e, hogy ha $lf(S_1,R) = lf(S_2,R)$, akkor $lf(S_1 \cup S_2, R) = lf(S_1,R) \vee lf(S_2,R)$?

3.5. Igaz-e, ha $\forall x,y \in A : \; x \in \lceil lf(S_1, \mathcal{P}(\{y\})) \rceil \Leftrightarrow x \in \lceil lf(S_2, \mathcal{P}(\{y\})) \rceil$, akkor $\mathcal{D}_{p(S_1)} = \mathcal{D}_{p(S_2)}$?

3.6. $S_1, S_2 \subseteq A \times A^{**}$ programok. Igaz-e, ha $\forall H : A \to \mathbb{L}$ esetén $lf(S_1,H) = lf(S_2,H)$, akkor S_1 ekvivalens S_2-vel?

3.7. $A = \mathbb{N}$. $S \subseteq \mathbb{N} \times \mathbb{N}^{**}$.

$$\begin{array}{l} S = \{(a, <a \cdots>) \mid a \equiv 1 \mod (4)\} \\ \quad \cup \{(b, <b>), (b, <b, b/2>) \mid b \equiv 2 \mod (4)\} \\ \quad \cup \{(c, <c, 2*c>) \mid c \equiv 3 \mod (4)\} \\ \quad \cup \{(d, <d, d/2>) \mid d \equiv 0 \mod (4)\} \end{array}$$

$H(x) = (x$páros szám$)$. $\lceil lf(S,H) \rceil =$?

3.8. Adott az $A = V \times V \times \mathbb{L}$ állapottér $(V = \{1,2,3\})$ és a $B = V \times V$ paramétertér, továbbá az F_1 és F_2 feladatok.

$$F_1 = \{((a_1,a_2,l),(b_1,b_2,k)) \mid k = (a_1 > a_2)\},$$

F_2 specifikációja pedig:

$$\begin{array}{lccccc} A = & V & \times & V & \times & \mathbb{L} \\ & a_1 & & a_2 & & l \end{array}$$

$$\begin{array}{lccc} B = & V & \times & V \\ & a_1' & & a_2' \end{array}$$

$Q : (a_1 = a_1' \wedge a_2 = a_2')$
$R : (Q \wedge l = (a_1' > a_2'))$

Azonosak-e az F_1 és F_2 feladatok?

3.9. Tekintsük az alábbi két feladatot: F_1 specifikációja:

$$\begin{array}{rl} A = & \mathbb{Z} \times \mathbb{Z} \\ & x \quad y \end{array}$$

$$\begin{array}{rl} B = & \mathbb{Z} \\ & x' \end{array}$$

$Q : (x = x')$
$R : (Q \wedge x = |y * y|)$

$$F_2 = \{((a, b), (c, d)) \mid c = a \wedge |d| * d = c\}.$$

Megadható-e valamilyen összefüggés F_1 és F_2 között?

3.10. Írd le szövegesen az alábbi feladatot: legyen $f : \mathbb{Z} \to \mathbb{Z}$,

$$\begin{array}{rl} A = & \mathbb{Z} \times \mathbb{Z} \times \mathbb{N}_0 \\ & m \quad n \quad l \end{array}$$

$$\begin{array}{rl} B = & \mathbb{Z} \times \mathbb{Z} \\ & m' \quad n' \end{array}$$

$Q : (m = m' \wedge n = n' \wedge m \leq n)$
$R : (Q \wedge l = \sum_{i=1}^{n} g(i))$

ahol $g : \mathbb{Z} \to \{0, 1\}$,

$$g(i) = \begin{cases} 1, & \text{ha } \exists x \in \mathbb{Z} : \ (f(i) = x \wedge \forall j \in [m..n] : f(j) \leq m) \\ 0, & \text{különben} \end{cases}$$

3.11. Igaz-e a specifikáció tételének megfordítása? (Ha S megoldja F-et, akkor $\forall b \in B : \ Q_b \Rightarrow lf(S, R_b)$)

3.12. Tekintsük az alábbi feladatot:

$$\begin{array}{rl} A = & \mathbb{Z} \times \mathbb{Z} \\ & k \quad p \end{array}$$

$$\begin{array}{rl} B = & \mathbb{Z} \\ & k' \end{array}$$

$Q : (k = k' \wedge 0 < k)$
$R : (Q \wedge prim(p) \wedge \forall i > 1 : prim(i) \to |k - i| \geq |k - p|)$

ahol $prim(x) = (x$prímszám$)$.

Mit rendel a fent specifikált feladat az $a = (10, 1)$ és a $b = (9, 5)$ pontokhoz? Fogalmazd meg szavakban a feladatot!

3.13. $A = \underset{x}{\mathbb{N}} \times \underset{y}{\mathbb{N}} \times \underset{z}{\mathbb{N}} \quad B = \underset{x'}{\mathbb{N}} \times \underset{y'}{\mathbb{N}}$

$F_1, F_2 \subseteq A \times A$

F_1 specifikációja:

$$Q = (x = x' \wedge y = y')$$
$$R = (x = x' \wedge y = y' \wedge x'|z \wedge y'|z \wedge \forall j \in \mathbb{N} : \ (x'|j \wedge y'|j) \Rightarrow z|j)$$
$$F_2 = \{((a,b,c),(d,e,f)) \mid a = d \text{ és } b = e \text{ és } f|a * b \text{ és } a|f \text{ és } b|f\}$$

Megadható-e valamilyen összefüggés F_1 és F_2 között?

3.14. Adott egy $f : \mathbb{Z} \to \mathbb{Z}$ függvény.

$A = \underset{m}{\mathbb{Z}} \times \underset{n}{\mathbb{Z}} \times \underset{i}{\mathbb{Z}} \quad B = \underset{m'}{\mathbb{Z}} \times \underset{n'}{\mathbb{Z}}$

$F_1, F_2 \subseteq A \times A$

F_1 specifikációja:

$$Q = (m = m' \wedge n = n')$$
$$R = (m = m' \wedge n = n' \wedge i \in [m,n] \wedge \forall j \in [m,i) : \ f(j) < f(i) \wedge \forall j \in [i,n] : \ f(j) \leq f(i))$$

F_2 specifikációja:

$$Q = (m = m' \wedge n = n')$$
$$R = (i \in [m',n'] \wedge \forall j \in [m',n'] : \ f(j) \leq f(i)).$$

Azonos-e a két feladat?

3.15. Specifikáljuk a következő feladatot: $A = \mathbb{N}$ és $v : \mathbb{N} \to \{0,1\}$.

$F \subseteq A \times A$, $F = \{(s,s') \mid s' = \sum_{k=1}^{n} v(k)\}$

4. fejezet

Kiterjesztések

Az előző fejezetekben bevezetük a program és a feladatat fogalmát, és defi niáltuk az azonos állapottéren levő feladat–program párok között a megoldás fogalmát. A gyakorlatban általában azonban a feladat és a program különböző állapottéren van: példaként megemlíthetjük azt az esetet, amikor egy feladat megoldására a programban további változókat kell bevezetni, azaz a feladat állapotterét újabb komponensekkel kell bővíteni.

A továbbiakban megvizsgáljuk, hogy mit tudunk mondani a különböző állapottéren adott programok és feladatok viszonyáról a megoldás szempontjából és ennek alapján általánosítjuk (kiterjesztjük) a megoldás fogalmát erre az esetre is.

4.1. A feladat kiterjesztése

Ha egy feladat állapotterét kibővítjük újabb komponensekkel, mit jelentsen ez a feladat vonatkozásában? Elég kézenfekvő, hogy ebben az esetben a feladat ne tartalmazzon semmiféle kikötést az új komponensekre.

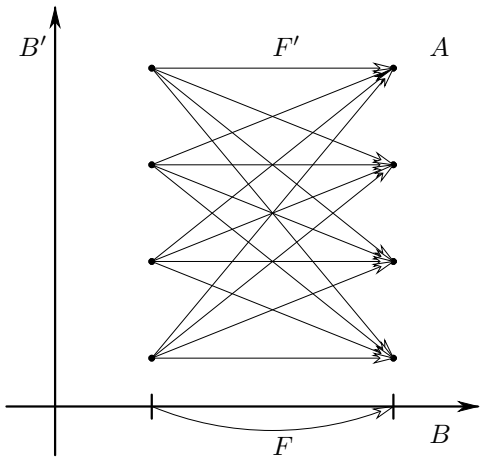
4.1. Definíció (FELADAT KITERJESZTÉSE). *Legyen a B állapottér altere az A állapottérnek. Az $F' \subseteq A \times A$ relációt az $F \subseteq B \times B$ feladat* kiterjesztésének *nevezzük, ha*

$$F' = \{(x, y) \in A \times A \mid (pr_B(x), pr_B(y)) \in F\}.$$

A defi níciót úgy is fogalmazhatjuk, hogy a feladat kiterjesztése az összes olyan $A \times A$-beli pontot tartalmazza, aminek B-re vett projekciója benne van F-ben. semmilyen megszorítást.

4.2. A program kiterjesztése

A program kiterjesztésének defi níciójában az új komponensekre azt a kikötést tesszük, hogy azok nem változnak meg a kiterjesztett programban. Ezzel azt a gyakorlati követelményt írjuk le, hogy azok a változók, amelyeket a program nem használ, nem változnak meg a program futása során.

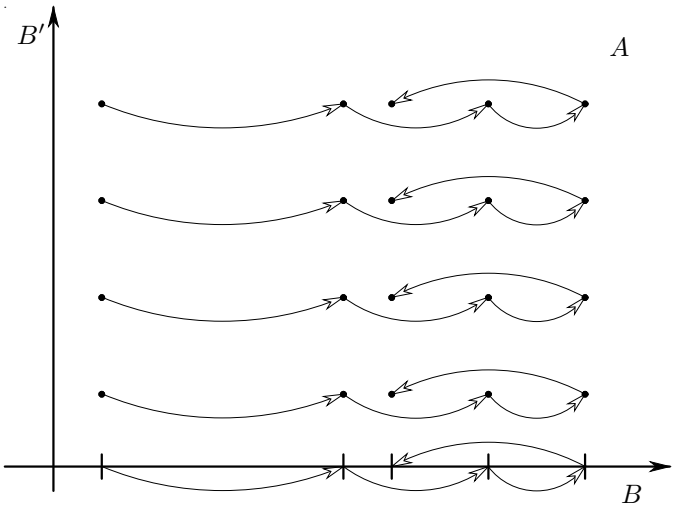


4.1. ábra. Feladat kiterjesztése

4.2. Definíció (PROGRAM KITERJESZTÉSE). *Legyen a B állapottér altere az A állapottérnek, és jelölje B' a B kiegészítő alterét A-ra. Legyen továbbá S program a B állapottéren. Ekkor az S' A-beli relációt az S program* kiterjesztésének *nevezzük, ha* $\forall a \in A$:

$$S'(a) = \{\alpha \in A^{**} \mid pr_B(\alpha) \in S(pr_B(a)) \wedge \forall i \in D_\alpha : pr_{B'}(\alpha_i) = pr_{B'}(a)\}$$

A fenti definíció alapján a kiterjesztett program értékkészletében csak olyan sorozatok vannak, amelyek „párhuzamosak” valamely sorozattal az eredeti program értékkészletéből.



4.2. ábra. Program kiterjesztése

Vajon a kiterjesztés megtartja a program-tulajdonságot? Erre a kérdésre válaszol az alábbi állítás.

4.2. állítás: *Legyen a B állapottér altere az A állapottérnek, és jelölje B' a B kie-*

gészítő alterét A-ra. Legyen továbbá S program a B állapottéren, és S' az S kiterjesztése A-ra. Ekkor S' program.

A tétel bizonyítása rendkívül egyszerű, a feladatok között szerepel.

A program kiterjesztése lehetőséget ad az ekvivales programok fogalmának kiterjesztésére is.

4.3. Definíció (PROGRAMOK EKVIVALENCIÁJA). *Legyenek $S_1 \subseteq A_1 \times A_1^{**}$, $S_2 \subseteq A_2 \times A_2^{**}$ programok, B altere mind A_1-nek, mind A_2-nek. Azt mondjuk, hogy S_1* ekvivalens *S_2-vel B-n,*

$$pr_B(p(S_1)) = pr_B(p(S_2)).$$

A definíciónak egyszerű következménye az is, hogy a két ekvivalens program a közös altéren pontosan ugyanazokat a feladatokat oldja meg.

Valójában attól, hogy két program ekvivalens – azaz megegyezik a programfüggvényük – egyéb tulajdonságaik nagyon eltérők lehetnek. Ilyen – nem elhanyalgolható – különbség lehet például a hatékonyságukban. Egyáltalán nem mindegy, hogy egy program mennyi ideig fut és mekkora memóriára van szüksége. A program ezen jellemzőinek vizsgálatával azonban itt nem foglalkozunk.

A definíciókból közvetlenül adódik a következő állítás:

4.3. állítás: *Egy program kiterjesztése és az eredeti program az eredeti állapottéren ekvivalens.*

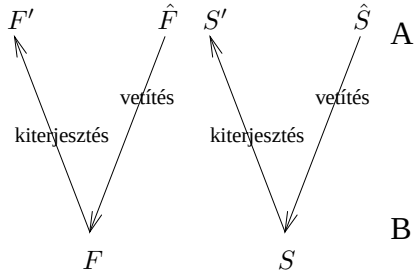
4.3. Kiterjesztési tételek

Az alábbiakban következő tételcsoport a megoldás és a kiterjesztések közötti kapcsolatot vizsgálja.

Ha van egy A állapottér, aminek B altere, a B-n definiált feladatok és programok megfelelői A-n, a feladatok és programok kiterjesztései A-ra. Az A-n definiált feladat megfelelőjének a B-n tekinthetjük a feladat vetületét B-re. A programok esetében ez közvetlenül nem alkalmazható, mivel a program vetülete nem biztos, hogy program. Nem biztos hogy a sorozatok redukáltak. Természetesen, ha $\hat{S} \subseteq A \times A^{**}$ program, akkor az

$$S = \{(b, \beta) \in B \times B^{**} \mid (a, \alpha) \in \hat{S} \text{ és } b = pr_B(a) \text{ és } \beta = red(pr_B(\alpha))\}$$

már program és a B állapottéren S és $\hat{S}$ ekvivalens. Tehát egy $\hat{S} \subseteq A \times A^{**}$ programhoz mindíg található olyan $S \subseteq B \times B^{**}$ program, ami vele ekvivalens B-n.



4.3. ábra. Kapcsolat A és B között.

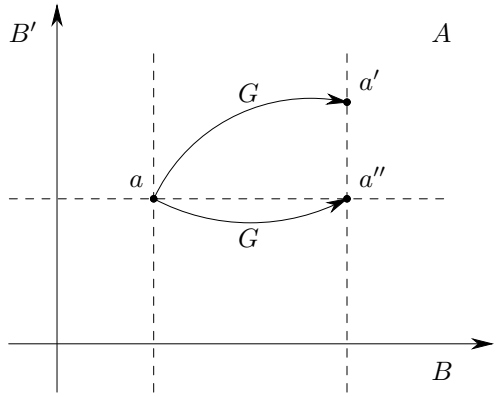
Ilyen módon, ahogy 4.3 ábra is mutatja, a kiterjesztés és a vetítés segítsével kapcsolatot létesítünk az A és B állapottereken defi niált programok között.

Természetesen általában sok olyan feladat van A-n, aminek a vetülete F, ilyen például az F kiterjesztése, de nem csak az. Tehát az 4.3 ábrán fölfelé mutató nyilak injektív megfeleltetések, a lefelé mutatók pedig szürjektívek.

Megjegyezzük még, hogy $\hat{F}$, vagyis egy olyan feladat, aminek a vetülete F, mindíg része F kiterjesztésének. Ugyanez a programok(programfüggvények) esetében nem igaz.

Megvizsgáljuk, hogy milyen esetekben következtethetünk az A állapottéren fennálló megoldásból, ugyanerre a B állapottéren és fordítva. Ahhoz, hogy a feltételeket megfogalmazhassuk szükségünk lesz néhány defi nícióra.

4.4. Definíció (BŐVÍTETT IDENTITÁS). *Legyen B altere A-nak, B' a B kiegészítő altere A-ra, $G \subseteq A \times A$ feladat. A G* bővített identitás *B' felett, ha $\forall (a, a') \in G$: $\exists a'' \in A$, hogy $(a, a'') \in G \wedge pr_{B'}(a) = pr_{B'}(a'') \wedge pr_B(a') = pr_B(a'')$.*



4.4. ábra. Bővített identitás

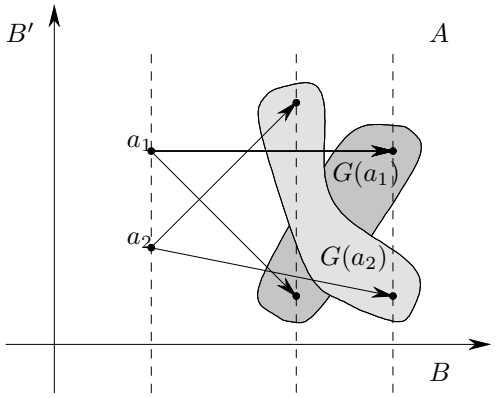
Ha egy feladat bővített identitás, az azt jelenti, hogy a feladat "megengedi", hogy a kiegészítő altérbeli komponensek változatlanok maradjanak. Könnyű látni a defi níciók alapján, hogy egy feladat kiterjesztése is és egy program kiterjesztésének a programfüggvénye is bővített identitás.

4.5. Definíció (VETÍTÉSTARTÁS). *Legyen B altere A-nak, $G \subseteq A \times A$ feladat. A G* vetítéstartó *B felett, ha $\forall a_1, a_2 \in \mathcal{D}_G$: $(pr_B(a_1) = pr_B(a_2)) \Rightarrow (pr_B(G(a_1)) = pr_B(G(a_2)))$.*

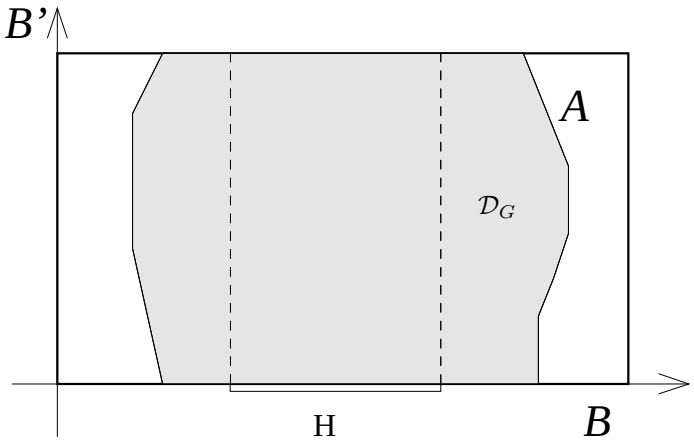
A vetítéstartás nem jelenti azt, hogy a reláció nem nem függ a kiegészítő altér komponenseitől, hiszen mint az 4.5 ábra mutatja, két azonos vetületű pont képe lehet kűlönböző, csak a vetületük azonos. Ebben az esetben is igaz, hogy egy feladat kiterjesztése vetítéstartó(ebben az esetben a képek is megegyeznek) és a program kiterjesztése, és így a kiterjesztés programfüggvénye is vetítéstartó.

4.6. Definíció (FÉLKITERJESZTÉS). *Legyen B altere A-nak, $G \subseteq A \times A$ feladat, $H \subseteq B$. Azt mondjuk, hogy a G* félkiterjesztés *H felett, ha $pr_B^{-1}(H) \subseteq \mathcal{D}_G$.*

A félkiterjesztés szemléletes jelentése, hogy s kiégészítő altér felől nézve az értelmezési tartományban nincsenek "lyukak". Most is igaz, hogy egy feladat kiterjesztése a feladat értelmezési tartománya fölött félkiterjesztés. Ugyancsak igaz, hogy a program kiterjesztésének programfüggvénye az eredeti programfüggvény értelmezési tartománya fölött félkiterjesztés.



4.5. ábra. Vetítéstartás



4.6. ábra. Félkiterjesztés

Az imént bevezetett definíciók segítségével kimondhatók azok az állítások, amelyek a kiterjesztések és a projekció valamint a megoldás közötti kapcsolatot vizsgáló tételcsoportot alkotják. A jelölések az 4.3 ábrának megfelelőek.

4.3. TÉTEL: KITERJESZTÉSI 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 a kiterjesztése A-ra. Legyen továbbá $\hat{F} \subseteq A \times A$ olyan feladat, melyre $pr_B(\hat{F}) = F$, és $\hat{S} \subseteq A \times A^{**}$ pedig olyan program, amely ekvivalens S-sel B-n. Ekkor az alábbi állítások teljesülnek:

(1) ha S' megoldása F'-nek, akkor S megoldása F-nek,

(2) ha S' megoldása $\hat{F}$-nek, akkor S megoldása F-nek,

(3) ha $\hat{S}$ megoldása F'-nek, akkor S megoldása F-nek,

(4) a. ha $\hat{S}$ megoldása $\hat{F}$-nek és $p(\hat{S})$ vetítéstartó B felett, akkor S megoldása F-nek,

b. ha $\hat{S}$ megoldása $\hat{F}$-nek és $\hat{F}$ félkiterjesztés $\mathcal{D}_F$ felett, akkor S megoldása F-nek,

(5) ha S megoldása F-nek, akkor S' megoldása F'-nek,

(6) ha S megoldása F-nek és $\hat{F}$ bővített identitás B' felett és vetítéstartó B felett, akkor S' megoldása $\hat{F}$-nek,

(7) ha S megoldása F-nek és $p(\hat{S})$ félkiterjesztés $\mathcal{D}_F$ felett, akkor $\hat{S}$ megoldása F'-nek.

Bizonyítás: Mielőtt sorra bizonyítanánk az egyes tételeket, vegyük észre, hogy a (4) tételből következik az első három, hiszen S' ekvivalens S-sel B-n és $p(S')$ vetítéstartó, illetve $pr_B(F') = F$ és F' félkiterjesztés $\mathcal{D}_F$-en. Hasonló meggondolások alapján a (6) tételből is következik az (5) tétel, hiszen F' bővített identitás B' felett és vetítéstartó B felett. Elegendő tehát a (4), (6), és (7) tételeket bizonyítani.

Tekintsük először a (4) tétel bizonyítását: Legyen $b \in \mathcal{D}_F$ tetszőleges. Ekkor

$$\begin{aligned} b \in \mathcal{D}_F \quad &\Rightarrow \quad \exists a \in \mathcal{D}_{\hat{F}} : pr_B(a) = b \\ &\overset{\text{megoldás}}{\Rightarrow} \quad a \in \mathcal{D}_{p(\hat{S})} \\ &\overset{\hat{S} \text{ ekv. } S}{\Rightarrow} \quad pr_B(a) \in \mathcal{D}_{p(S)} \end{aligned}$$

tehát $\mathcal{D}_F \subseteq \mathcal{D}_{p(S)}$, s így a megoldás első kritériumának teljesülését bebizonyítottuk. Tekintsük most a második kritériumot: legyen $b \in \mathcal{D}_F$ tetszőlegesen rögzített. Ekkor

$$\begin{aligned} p(S)(b) &= \bigcup_{a \in pr_B^{-1}(b) \cap \mathcal{D}p(\hat{S})} pr_B(p(\hat{S})(a)) \\ F(b) &= \bigcup_{a \in pr_B^{-1}(b) \cap \mathcal{D}\hat{F}} pr_B(\hat{F}(a)) \end{aligned}$$

Az a. esetben, azaz ha $p(\hat{S})$ vetítéstartó, akkor $\forall x, y \in pr_B^{-1}(b) \cap \mathcal{D}_{p(\hat{S})}$-ra $pr_B(p(\hat{S})(x)) = pr_B(p(\hat{S})(y))$. Ekkor tetszőleges $a \in pr_B^{-1}(b) \cap \mathcal{D}_{\hat{F}} pr_B(\hat{F}(a))$ esetén, mivel a megoldás definíciója miatt $p(\hat{S}(a)) \subseteq \hat{F}(a)$,

$$p(S)(b) = pr_B(p(\hat{S})(a)) \subseteq pr_B(\hat{F}(a)) \subseteq F(b).$$

Tehát a megoldás második feltétele is teljesül.

A b. esetben, azaz ha $\hat{F}$ félkiterjesztés, akkor $pr_B^{-1}(b) \subseteq \mathcal{D}_{\hat{F}}$, azaz $a \in pr_B^{-1}(b) \cap \mathcal{D}_{\hat{F}} = pr_B^{-1}(b)$ és a megoldás definíciója miatt

$$\forall a \in pr_B^{-1}(b) : p(\hat{S})(a) \subseteq \hat{F}(a)$$

tehát

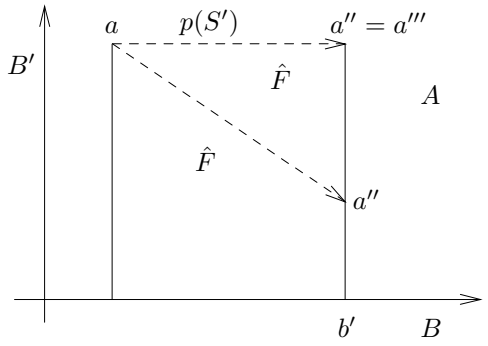
$$\begin{aligned} & \bigcup_{a \in pr_B^{-1}(b)} p(\hat{S})(a) \subseteq \bigcup_{a \in pr_B^{-1}(b)} \hat{F}(a) \\ \Rightarrow \quad & pr_B\Big(\bigcup_{pr_B^{-1}(b)} p(\hat{S})(a) \Big) \subseteq pr_B\Big(\bigcup_{pr_B^{-1}(a)} \hat{F}(a) \Big) \\ \Rightarrow \quad & \bigcup_{pr_B^{-1}(a)} pr_B(p(\hat{S})(a)) \subseteq \bigcup_{pr_B^{-1}(a)} pr_B(\hat{F}(a)) \\ \Rightarrow \quad & p(S)(b) \subseteq F(b) \end{aligned}$$

és ezzel ebben az esetben is beláttuk, hogy az S program megoldja az F feladatot. Nézzük most a (6) tétel bizonyítását.

1. Először belátjuk, hogy $\mathcal{D}_{\hat{F}} \subseteq \mathcal{D}_{p(S')}$.

   Legyen $a \in \mathcal{D}_{\hat{F}}$. Ekkor $pr_B(a) \in \mathcal{D}_F$. Felhasználva, hogy S megoldása F-nek, $pr_B(a) \in \mathcal{D}_{p(S)}$. A program kiterjesztésének definíciójából következik, hogy ekkor $a \in \mathcal{D}_{p(S')}$.

2. Ezután megmutatjuk, hogy $\forall a \in \mathcal{D}_{\hat{F}} : p(S')(a) \subseteq \hat{F}(a)$ is teljesül.



4.7. ábra.

   Az 4.7 ábrának megfelelően legyen $a \in \mathcal{D}_{\hat{F}}$ és $a' \in p(S')(a)$. Ekkor – felhasználva, hogy S' az S kiterjesztése – a'-re fennáll az alábbi tulajdonság:

$$pr_{B'}(a') = pr_{B'}(a)$$

   Legyen $b' = pr_B(a')$. Ekkor $b' \in p(S)(pr_B(a))$. Mivel S megodja F-et, adódik, hogy $b' \in F(pr_B(a))$. Ekkor – mivel $\hat{F}$ vetítéstartó B felett és F a $\hat{F}$ projekciója – adódik, hogy $\exists a'' \in \hat{F}(a) : pr_B(a'') = b'$. Felhasználva, hogy $\hat{F}$ bővített identitás B' felett, $\exists a''' \in \hat{F}(a)$, amelyre

$$pr_{B'}(a''') = pr_{B'}(a) \text{ és } pr_B(a''') = b'.$$

   Ekkor viszont $a' = a'''$, azaz $a' \in \hat{F}(a)$.

Most már csak a (7) állítás bizonyítása van hátra:

(1) Legyen $a \in \mathcal{D}_{F'}$. Ekkor a feladat kiterjesztése definíciója alapján $pr_B(a) \in \mathcal{D}_F$. Mivel $p(\hat{S})$ félkiterjesztés $\mathcal{D}_F$ felett, $a \in \mathcal{D}_{p(\hat{S})}$.

(2) Legyen $a \in \mathcal{D}_{F'}$, $a' \in p(S')(a)$ és $b' = pr_B(a')$. Ekkor $b' \in p(S)(pr_B(a))$, hiszen $p(S)$ az $\hat{S}$ vetülete. Mivel S megodja F-et, adódik, hogy $b' \in F(pr_B(a))$, de a feladat kiterjesztésének definíciója alapján $\forall x \in pr_B^{-1}(b') : x \in F'(a)$, így $b' \in F'(a)$. Tehát ba megoldás második feltétele is teljesül.

Ezzel a (7) állítást is bebizonyítottuk. □

4.4. A megoldás fogalmának kiterjesztése

A kiterjesztési tételek alapján általánosítjuk a megoldás fogalmát. Az eredeti definícióban kikötöttük, hogy a feladat és a program állapottere azonos, erre a feltételre valójában nincs szükség, elég ha közös állapottére kiterjesztve a feladatot és a programot teljesölnek a megoldás feltételei.

4.7. Definíció (A MEGOLDÁS KITERJESZTÉSE). *Legyenek H egy tetszőleges halmaz, $A_h, h \in H$ legfeljebb megszámlálható halmazok és I és J véges részhalmazai H-nak. $A = \overset{\times}{_{i \in I}} A_i$ és $B = \overset{\times}{_{j \in J}} A_j$. $F \subseteq A \times A$ feladat és $S \subseteq B \times B^{**}$ program. Ha létezik C állapottér, aminek A és B is altere és S kiterjesztése C-re megoldása F C-re való kiterjesztettjének, akkor S megoldása F-nek.*

A kiterjesztési tételekből, méghozzá az 1-ből és az 5-ből azonnal adódik, hogy a definícióban a "létezik" szót "minden"re cserélhetnénk. Az is nyílvánvaló, hogy "közös többszörös" tulajdonságú állapottér, vagyis olyan, aminek A is és B is altere, mindíg létezik: $\overset{\times}{_{k \in I \cup J}} A_k$.

Ezért a definíciót úgy is fogalmazhatjuk, hogy a $C = \overset{\times}{_{k \in I \cup J}} A_k$ allapottérre kiterjesztve a feladatot és a programot, teljesülnek a megoldás feltételei.

A kiterjesztett megoldás fogalom ismeretében érdemes a kiterjesztési tételeket újra megvizsgálni. Az 1. és 5. tétel jelentőségét a definíciónál már tárgyaltuk. A 2., illetve 3. tétel azt jelenti, hogy ha egy program megoldása egy feladatnak akkor akár a program, akár a feladat állapoterén is teljesülnek a megoldás feltételei. Ugyanez fordítva már csak bizonyos feltételek teljesülése esetén igaz (5., 6. tétel).

4.5. A feladat kiterjesztése és a specifikáció tétele

Emlékeztetünk a specifikáció tételének megfelelő formában megadott – állapottér, paramétertér, elő- és utófeltételek – feladatokra. Példaként felírtuk két szám összegét:

$$
\begin{array}{l}
A = \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} \\
\quad\;\; x \quad y \quad z \\
B = \mathbb{Z} \times \mathbb{Z} \\
\quad\;\; x' \quad y' \\
Q : (x = x' \wedge y = y') \\
R : (z = x' + y')
\end{array}
$$

Mi lesz ennek e feladatnak a kiterjesztése egy $A' = \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} \times \mathbb{N} \times \mathbb{L}$ állapottérre? A válasz első pillantásra meglepő.

$$
\begin{array}{l}
A = \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z} \times \mathbb{N} \times \mathbb{L} \\
\quad\;\; x \quad y \quad z \quad u \quad v \\
B = \mathbb{Z} \times \mathbb{Z} \\
\quad\;\; x' \quad y' \\
Q : (x = x' \wedge y = y') \\
R : (z = x' + y')
\end{array}
$$

Általánosságban is igaz, hogy a feladat kiterjesztésének specifikációja, a kibővített állapottértől eltekintve, megegyezik az eredeti feladat specifikációjával.

4.6. Példák

4.1. példa: $A = \{1, 2, 3\}, B = A \times \{1, 2, 3\}$. $F \subseteq A \times A$. $F = \{(1, 2), (1, 3)\}$. Mi az F kiterjesztettje B-re?

Megoldás: A feladat kiterjesztésének definíciója alapján:

$$\begin{array}{lllll} F = \{ & ((1,1),(2,1)), & ((1,1),(2,2)), & ((1,1),(2,3)), & ((1,2),(2,1)), \\ & ((1,2),(2,2)), & ((1,2),(2,3)), & ((1,3),(2,1)), & ((1,3),(2,2)), \\ & ((1,3),(2,3)), & ((1,1),(3,1)), & ((1,1),(3,2)), & ((1,1),(3,3)), \\ & ((1,2),(3,1)), & ((1,2),(3,2)), & ((1,2),(3,3)), & ((1,3),(3,1)), \\ & ((1,3),(3,2)), & ((1,3),(3,3)) & \} & \end{array}$$

4.2. példa: Adott az $\mathbb{L}\times\mathbb{L}$ állapottéren az $F = \{((l,k),(l',k')) \mid k' = (l \wedge k)\}$ feladat, és az $A' = \mathbb{L}\times\mathbb{L}\times V$ állapottéren ($V = \{1,2\}$) a következő program:

$$\begin{array}{lll} S = \{ & (ii1, \langle ii1, ih2, hi2\rangle), & (ii2, \langle ii2, hh1, ii1\rangle), \\ & (ii2, \langle ii2, ih2, hi1, hi2\rangle), & (ih1, \langle ih1\rangle), \\ & (ih2, \langle ih2, ii1, hh1\rangle), & (hi1, \langle hi1, hh2\rangle), \\ & (hi2, \langle hi2, hi1, ih1\rangle), & (hi2, \langle hi2, hh1, hh2\rangle), \\ & (hh1, \langle hh1, ih1\rangle), & (hh2, \langle hh2\rangle) \qquad \} \end{array}$$

Megoldja-e S az F A'-re való kiterjesztettjét?
Megoldás: Írjuk fel az F A'-re való kiterjesztettjét:

$$\begin{array}{llllll} F' = \{ & (ii1,ii1), & (ii1,hi1), & (ii1,ii2), & (ii1,hi2), & \\ & (ii2,ii1), & (ii2,hi1), & (ii2,ii2), & (ii2,hi2), & \\ & (ih1,ih1), & (ih1,hh1), & (ih1,ih2), & (ih1,hh2), & \\ & (ih2,ih1), & (ih2,hh1), & (ih2,ih2), & (ih2,hh2), & \\ & (hi1,ih1), & (hi1,hh1), & (hi1,ih2), & (hi1,hh2), & \\ & (hi2,ih1), & (hi2,hh1), & (hi2,ih2), & (hi2,hh2), & \\ & (hh1,ih1), & (hh1,hh1), & (hh1,ih2), & (hh1,hh2), & \\ & (hh2,ih1), & (hh2,hh1), & (hh2,ih2), & (hh2,hh2) & \} \end{array}$$

Az S program programfüggvénye:

$$\begin{array}{llll} p(S) = \{ & (ii1,hi2), & (ii2,ii1), & (ii2,hi2), & (ih1,ih1), \\ & (ih2,hh1), & (hi1,hh2), & (hi2,ih1), & (hi2,hh2), \\ & (hh1,ih1), & (hh2,hh2) & \} & \end{array}$$

A megoldás definíciója két pontjának teljesülését kell belátnunk. $\mathcal{D}_{F'} \subseteq= \mathcal{D}_{p(S)}$ triviálisan teljesül, hiszen mindkét halmaz a teljes állapottér. Vizsgáljuk meg most, hogy $\forall a \in \mathcal{D}_F : p(S)(a) \subseteq F'(a)$ teljesül-e!

$$\begin{array}{rcl} p(S)(ii1) = \{hi2\} & \subseteq & \{ii1, hi1, ii2, hi2\} = F(ii1) \\ p(S)(ii2) = \{ii1, hi2\} & \subseteq & \{ii1, hi1, ii2, hi2\} = F(ii2) \\ p(S)(ih1) = \{ih1\} & \subseteq & \{ih1, hh1, ih2, hh2\} = F(ih1) \\ p(S)(ih2) = \{hh1\} & \subseteq & \{ih1, hh1, ih2, hh2\} = F(ih2) \\ p(S)(hi1) = \{hh2\} & \subseteq & \{ih1, hh1, ih2, hh2\} = F(hi1) \\ p(S)(hi2) = \{ih1, hh2\} & \subseteq & \{ih1, hh1, ih2, hh2\} = F(hi2) \\ p(S)(hh1) = \{ih1\} & \subseteq & \{ih1, hh1, ih2, hh2\} = F(hh1) \\ p(S)(hh2) = \{hh2\} & \subseteq & \{ih1, hh1, ih2, hh2\} = F(hh2) \end{array}$$

Tehát az S program megoldja az F feladat kiterjesztettjét.

4.3. példa: Igaz-e, ha $S \subseteq B \times B$, A altere B-nek, akkor

$$\mathcal{D}_{pr_A(p(S))} = pr_A(\mathcal{D}_{p(S)})?$$

Megoldás: Próbáljuk meg az állítást kétirányú tartalmazkodás belátásával bizonyítani. $\mathcal{D}_{pr_A(p(S))} \subseteq pr_A(\mathcal{D}_{p(S)})$: Legyen $a \in \mathcal{D}_{pr_A(p(S))}$. Ekkor

$$\begin{array}{ll} & \exists a' \in A : (a, a') \in pr_A(p(S)) \\ \Rightarrow & \exists (b, b') \in p(S) : pr_A(b, b') = (a, a') \\ \Rightarrow & b \in \mathcal{D}_{p(S)} \Rightarrow pr_A(b) = a \in pr_A(\mathcal{D}p(S)). \end{array}$$

$pr_A(\mathcal{D}_{p(S)}) \subseteq \mathcal{D}_{pr_A(p(S))}$: Legyen $a \in pr_A(\mathcal{D}_{p(S)})$. Ekkor

$$\begin{array}{ll} & \exists b \in \mathcal{D}_{p(S)} : pr_A(b) = a \\ \Rightarrow & \exists b' \in B : (b, b') \in p(S) \\ \Rightarrow & (a, pr_A(b')) \in pr_A(p(S)) \\ \Rightarrow & a \in \mathcal{D}_{pr_A(p(S))} \end{array}$$

és ezzel az állítást bebizonyítottuk.

4.7. Feladatok

4.1. $A = \mathbb{N}, B = A \times \mathbb{N}$. $F \subseteq A \times A$. $F = \{(q, r) \mid r = q + 1\}$. Mi az F kiterjesztettje B-re?

4.2. Igaz-e, ha $S \subseteq B \times B^{**}$ program, A altere B-nek, akkor S $A \times A$-ra történő projekciójának kiterjesztése B-re azonos S-sel?

4.3. Bizonyítsuk be, hogy egy program kiterjesztettje valóban program!

4.4. $A = A_1 \times A_2 \times \cdots \times A_n$. Mondjunk példát olyan programra, amelynek egyetlen valódi altérre vett projekciója sem program. $(A_k = \mathbb{N}, k = 1, \ldots, n)$.

4.5. Legyen A altere B-nek, $F \subseteq A \times A$, $F'' \subseteq B \times B$, F' az F kiterjesztettje B-re. Igaz-e, hogy

a) ha $F = pr_A(F'')$, akkor F'' az F kiterjesztettje?

b) $F' = pr_A^{(-1)}(F)$? ill. $F' = pr_A^{-1}(F)$?

4.6. Legyen $F \subseteq A \times A$, $F' \subseteq B \times B$, $F'' \subseteq C \times C$, $F''' \subseteq D \times D$, ahol $B = A \times A_1$, $C = A \times A_2, D = A \times A_1 \times A_2$, és legyen F', F'', F''' az F kiterjesztése rendre B-re, C-re, D-re. Igaz-e, hogy F''' az F'' kiterjesztése D-re? Add meg az F' és az F'' közötti kapcsolatot a projekció és a kiterjesztés fogalmának segítségével!

4.7. B és C altere A-nak. $F \subseteq A \times A, F_1 \subseteq B \times B, F_2 \subseteq C \times C$. F_1 az F projekciója B-re. F az F_2 kiterjesztése A-ra. Igaz-e, hogy az F_1 feladat A-ra való kiterjesztettjének C-re vett projekciója megegyezik F_2-vel?

5. fejezet

A típus

Az állapottér definíciójában szereplő halmazokat típusértékhalmazoknak neveztük, és csak annyit mondtunk róluk, hogy legfeljebb megszámlálhatóak.

A továbbiakban arról lesz szó, hogy ezek a halmazok hogyan jönnek létre, milyen közös tulajdonság jellemző az elemeikre.

5.1. A típusspecifikáció

Először bevezetünk egy olyan fogalmat, amit arra használhatunk, hogy pontosan leírjuk a követelményeinket egy típusértékhalmazzal, és a rajta végezhető műveletekkel szemben.

5.1. Definíció (TÍPUSSPECIFIKÁCIÓ). *A $\mathcal{T}_s = (T, I_s, \mathbb{F})$ hármast* típusspecifikációnak *nevezzük, ha teljesülnek rá a következő feltételek:*

1. *T: tetszőleges alaphalmaz,*
2. *$I_s : T \to \mathbb{L}$ specifikációs invariáns,*
   $T_s = \lceil I_s \rceil$ a típusértékhalmaz,
3. *$\mathbb{F} = \{F_1, F_2, \ldots, F_n\}$, ahol $\forall i \in [1..n] : F_i \subseteq A_i \times A_i$, amelyre*
   $A_i = A_{i_1} \times \cdots \times A_{i_{n_i}}$ úgy, hogy $\exists j \in [1..n_i] : A_{i_j} = T$ és
   $\forall j \in [1..n_i] : (A_{i_j} = T) \Rightarrow pr_{A_{i_j}}(F_i) \subseteq T_s \times T_s$.

Vegyük észre, hogy az alaphalmaz és az invariáns tulajdonság segítségével azt fogalmazzuk meg, hogy mi az az értékhalmaz, aminek elemeivel foglalkozni akarunk, míg a feladatok halmazával azt írjuk le, hogy ezekre az elemekre milyen műveletek végezhetők el.

Az állapottér definíciójában szereplő típusértékhalmazok mind ilyen típusspecifikációban vannak definiálva. Az állapottér egy komponensét egy program csak a típusműveleteken keresztül változtathatja meg.

5.2. A típus

Vizsgáljuk meg, hogy a típusspecifikációban leírt követelményeket hogyan valósítjuk meg. Ehhez bevezetjük az elemi típusértékek halmazát, amit E-vel jelölünk.

5.2. Definíció (TÍPUS). *A $\mathcal{T} = (\varrho, I, \mathbb{S})$ hármast* típusnak *nevezzük, ha az alábbi feltételek teljesülnek rá:*

1. *$\varrho \subseteq E^* \times T$, reprezentációs függvény,*
2. *$I : E^* \to \mathbb{L}$, típusinvariáns tulajdonság,*
3. *$\mathbb{S} = \{S_1, S_2, \ldots, S_m\}$, ahol*
   $\forall i \in [1..m] : S_i \subseteq B_i \times B_i^{**}$ program, amelyre $B_i = B_{i_1} \times \cdots \times B_{i_{m_i}}$ úgy, hogy $\exists j \in [1..m_i] : B_{i_j} = E^*$ és $\nexists j \in [1..m_i] : B_{i_j} = T$.

A típus első két komponense az absztrakt adattípus reprezentációját írja le, míg a programhalmaz a típusműveletek implementációját tartalmazza.

A specifikáció és a típus kapcsolata

Meg kell még vizsgálnunk azt a kérdést, hogy mikor mondjuk, hogy egy típus megfelel a típusspecifikációnak, azaz a típus mikor teljesíti a specifikációban leírt követelményeket.

5.3. Definíció (MEGFELELTETÉS). *Egy $\mathcal{T} = (\varrho, I, \mathbb{S})$ típus* megfelel *a $\mathcal{T}_s = (T, I_s, \mathbb{F})$ típusspecifikációnak, ha*

1. *$\varrho([I]) = T_s$,*
2. *$\forall F \in \mathbb{F} : \exists S \in \mathbb{S} : S$ a ϱ-n keresztül megoldja F-et.*

Természetesen a megfelelés definíciója még nem teljes, meg kell még mondanunk, hogy mit értünk a ϱ-n keresztüli megoldáson. Ehhez először vizsgáljunk meg egy egyszerű esetet: tegyük fel, hogy a feladat és a program állapottere egykomponensű.

Persze a fenti definíciók figyelembe vételével ez azt jelenti, hogy a feladat állapottere T, a programé pedig E^*. Ekkor tulajdonképpen a ϱ-n keresztüli megoldást úgy kell elképzelni, mintha a megoldás definíciójában a programfüggvény $\varrho\big|_{\lceil I \rceil} \odot p(S) \odot \varrho\big|_{\lceil I \rceil}^{(-1)}$ reláció lenne, azaz

1. $\mathcal{D}_F \subseteq \mathcal{D}_{\varrho\big|_{\lceil I \rceil} \odot p(S) \odot \varrho\big|_{\lceil I \rceil}^{(-1)}}$, és
2. $\forall a \in \mathcal{D}_F : \varrho\big|_{\lceil I \rceil} \odot p(S) \odot \varrho\big|_{\lceil I \rceil}^{(-1)}(a) \subseteq F(a)$.

Legyen a továbbiakban $S \in \mathbb{S}$, és $F \in \mathbb{F}$, $F \subseteq A \times A$, $A = A_1 \times \cdots \times A_n$, $S \subseteq B \times B^{**}$, $B = B_1 \times \cdots \times B_n$. Azt mondjuk, hogy az B állapottér *illeszkedik* az A állapottérhez, ha

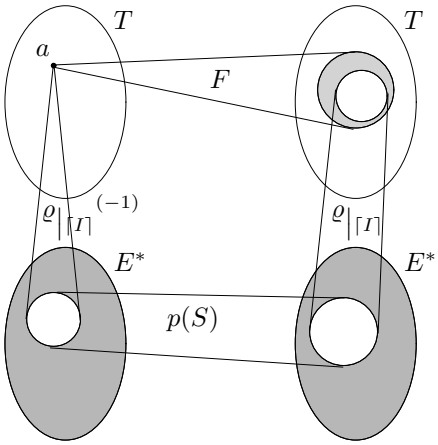
$$\forall i \in [1..n] : B_i = \begin{cases} E^*, & \text{ha } A_i = T \\ A_i, & \text{különben} \end{cases}$$

A fenti esetben legyen a $\gamma \subseteq B \times A$ leképezés az alábbi módon definiálva:

$$\forall b \in B : \gamma(b) = \gamma_1(b_1) \times \gamma_2(b_2) \times \ldots \gamma_n(b_n)$$

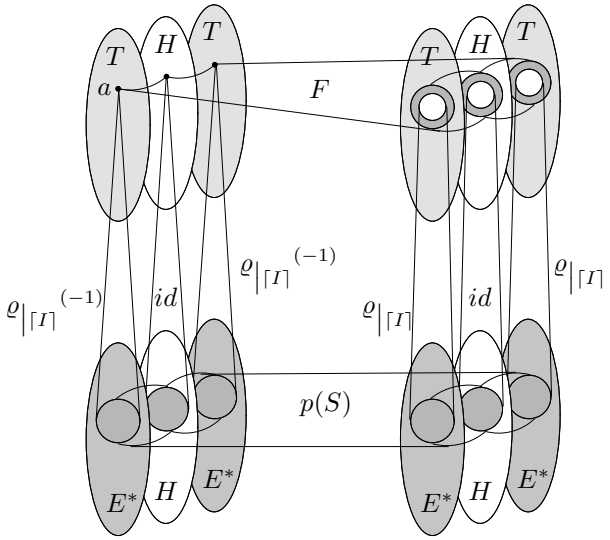
ahol $\forall i \in [1..n] : \gamma_i \subseteq B_i \times A_i$, és

$$\gamma_i = \begin{cases} \varrho\big|_{\lceil I \rceil}, & \text{ha } A_i = T \\ id_{A_i}, & \text{különben} \end{cases}$$



5.1. ábra. A ϱ-n keresztüli megoldás egykomponensű állapotterek között

A továbbiakban az ilyen felépítésű γ relációt $\gamma = (\gamma_1; \gamma_2; \ldots; \gamma_n)$-nel fogjuk jelölni. Vegyük észre, hogy a γ tulajdonképpen a ϱ egyfajta kiterjesztése több komponensű, de egymáshoz illeszkedő állapotterek esetén. Ezek után a megoldás defi nícióját felírhatjuk az ilyen esetre úgy, hogy az előző megoldásdefi nícióban $\varrho\big|_{\lceil I \rceil}$ helyére γ-t írunk.



5.2. ábra. A ϱ-n keresztüli megoldás illeszkedő állapotterek között

Most már csak egy kis lépés van hátra az általános eset leírásához: ha a program és a feladat állapottere nem illeszkedik egymáshoz, akkor tegyük illeszkedővé őket. Ez a kiterjesztés fogalmának felhasználásával könnyen megtehető.

5.4. Definíció (MEGOLDÁS ϱ-N KERESZTÜL). *Azt mondjuk, hogy az $S \subseteq B \times B^{**}$ program a ϱ-n keresztül* megoldja *az $F \subseteq A \times A$ feladatot, ha vannak olyan C és D illeszkedő terek, hogy A altere C-nek, B altere D-nek, és*

1. $\mathcal{D}_{F'} \subseteq \mathcal{D}_{\gamma \odot p(S') \odot \gamma^{(-1)}}$, *és*
2. $\forall a \in \mathcal{D}_{F'} : \gamma \odot p(S') \odot \gamma^{(-1)}(a) \subseteq F'(a)$,

ahol $\gamma \subseteq D \times C$ a fenti értelemben definiált leképezés, S' az S kiterjesztése D-re, F' pedig az F kiterjesztése C-re.

Természetesen egy típusspecifikációnak több különböző típus is megfelelhet. Ekkor az, hogy melyiket választjuk a reprezentáció és a műveletek implementációinak más további tulajdonságaitól – ilyen például a reprezentáció memóriaigénye, vagy az implementációk műveletigénye – függ. Ez a döntés mindig a megoldandó programozási feladat függvénye.

5.3. A típusspecifikáció tétele

A típusspecifikáció és a típus fogalmának bevezetésével tulajdonképpen a feladat fogalmát általánosítottuk, míg a megfeleltetés a megoldás fogalmának volt egyfajta általánosítása. Az imént megismert specifikáció tétele a megoldásra adott elégséges feltételt. Próbáljunk most a ϱ-n keresztüli megoldásra egy hasonló feltételt adni!

5.4. TÉTEL: TÍPUSPECIFIKÁCIÓ TÉTELE

Legyen $\mathcal{T}_s = (T, I_s, \mathbb{F})$ és $\mathcal{T} = (\varrho, I, \mathbb{S})$ adott típusspecifikáció és típus, és tegyük fel, hogy a reprezentáció helyes, azaz $\varrho(\lceil I \rceil) = \lceil I_s \rceil$. Legyen továbbá $F \in \mathbb{F}$, az F állapottere A, egy paramétertere B, elő és utófeltétele pedig Q_b és R_b. Legyen $S \in \mathbb{S}$ és tegyük fel, hogy S állapottere illeszkedik F állapotteréhez. Definiáljuk a következő állításokat:

$$\begin{aligned} \lceil Q_b^\gamma \rceil &= \lfloor Q_b \circ \gamma \rfloor \\ \lceil R_b^\gamma \rceil &= \lceil R_b \circ \gamma \rceil \end{aligned}$$

ahol γ a program és a feladat állapottere közötti, a ϱ-n keresztüli megoldás definíciójában szereplő leképezés. Ekkor ha $\forall b \in B : Q_b^\gamma \Rightarrow lf(S, R_b^\gamma)$, akkor az S program a ϱ-n keresztül megoldja az F feladatot.

Bizonyítás: A ϱ-n keresztüli megoldás definíciója két pontjának teljesülését kell belátnunk:

1. $\mathcal{D}_F \subseteq \mathcal{D}_{\gamma \odot p(S) \odot \gamma^{(-1)}}$, ui.

   Legyen $a \in \mathcal{D}_F$. Ekkor $\exists b \in B : a \in \lceil Q_b \rceil$. Mivel $\varrho(\lceil I \rceil) = \lceil I_s \rceil$,

   $$\gamma^{(-1)}(a) \neq \emptyset \wedge \gamma^{(-1)}(a) \subseteq \lceil Q_b^\gamma \rceil.$$

   Felhasználva, hogy $\lceil Q_b^\gamma \rceil \subseteq \lceil lf(S, R_b^\gamma) \rceil$:

   $$\gamma^{(-1)}(a) \subseteq \mathcal{D}_{p(S)} \wedge p(S)(\gamma^{(-1)}(a)) \subseteq \lceil R_b^\gamma \rceil.$$

   Mivel $\lceil R_b^\gamma \rceil = \lceil R_b \circ \gamma \rceil$,

   $$p(S)(\gamma^{(-1)}(a)) \subseteq \mathcal{D}_\gamma$$

   tehát

   $$a \in \mathcal{D}_{\gamma \odot p(S) \odot \gamma^{(-1)}}.$$

2. $\forall a \in \mathcal{D}_F : \gamma \odot p(S) \odot \gamma^{(-1)} \subseteq F(a)$, ui.

   Legyen $a \in \mathcal{D}_F$. A bizonyítás első részében leírt lépéseket folytatva: mivel $p(S)(\gamma^{(-1)}(a)) \subseteq \lceil R_b \circ \gamma \rceil$,

$$\gamma(p(S)(\gamma^{(-1)}(a))) \subseteq \lceil R_b \rceil \subseteq F(a).$$

□

Az, hogy a fenti tétel feltételei között kikötöttük, hogy a program állapottere illeszkedik a feladat állapotteréhez tulajdonképpen elhagyható. Ekkor a tétel a feladat és a program olyan kiterjesztéseire mondható ki, amelyek állapotterei illeszkednek egymáshoz (pontosan úgy, ahogy a megfeleltetést definiáltuk nem illeszkedő állapotterek között).

A következő példában megmutatjuk, hogy Q_b^γ-t gyenge igazsáhalmaz helyett erős igazsághalmazzal definiálnánk, akkor a tétel nem lenne igaz.

Legyen $\mathcal{T}_s = (T, I_s, \mathbb{F})$, $T = \{1, 2\}$, $I_s = \uparrow$, $\mathbb{F} = \{F\}$. Legyen F állapottere $A = T$ és a paramétertér is legyen ugyanez: $B = T$. Legyen F specifikációja:

$$\begin{array}{lllll} \lceil Q_1 \rceil & = & \{1\}, & \lceil R_1 \rceil & = & \{2\} \\ \lceil Q_2 \rceil & = & \emptyset, & \lceil R_2 \rceil & = & \{1\} \end{array}$$

Ekkor $\mathcal{D}_F = \{1\}$ és $F(1) = \{2\}$. Legyen továbbá az elemi értékek halmaza $E = \{a, b\}$, $\mathcal{T} = (\varrho, I, \mathbb{S})$, $\forall \alpha \in E^* : I(\alpha) = (|\alpha| = 1)$, és ϱ az egy hosszú sorozatokra:

$$\varrho(\langle a \rangle) = \varrho(\langle b \rangle) = \{1, 2\}.$$

Tegyük fel, hogy $\mathbb{S} = \{S\}$, $S \subseteq E^* \times (E^*)^{**}$, és rendelje S az állapottere minden pontjához az önmagából álló egy hosszú sorozatot. Ennek a programnak az állapottere illeszkedik a fenti feladat állapotteréhez, és $\gamma = \varrho$. Ekkor

$$\lceil Q_1 \circ \gamma \rceil = \emptyset \quad \text{és} \quad \lceil Q_2 \circ \gamma \rceil = \emptyset,$$

tehát

$$\begin{array}{lll} Q_1 \circ \gamma & \Rightarrow & lf(S, R_1 \circ \gamma) \\ Q_1 \circ \gamma & \Rightarrow & lf(S, R_1 \circ \gamma) \end{array}$$

Az viszont könnyen látható, hogy

$$\gamma \odot p(S) \odot \gamma^{(-1)}(1) = \{1, 2\} \not\subseteq F(1) = \{2\}$$

tehát a típus nem felel meg a specifikációnak.

5.4. Példák

5.1. példa: A típusértékek halmaza legyen a magyar abc magánhangzói! { a, á, e, é, i, í, o, ó, ö, ő, u, ú, ü, ű }. Szeretnénk tudni, hogy egy adott magánhangzónak melyik a (rövid ill. hosszú) párja. Legyen egy olyan típusműveletünk, amely erre a kérdésre választ tud adni. Az elemi típusértékek halmaza legyen a { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 } halmaz! Add meg a típusspecifikációt és készíts el egy olyan típust, ami megfelel a specifikációnak!

Megoldás: Írjuk fel először a típusspecifikációt! Legyen a magánhangzók halmaza MGH. Ekkor

$\mathcal{T}_s = (MGH, < igaz >, \{F\})$, ahol $F \subseteq MGH \times MGH$,

$F =$ { (a, á), (á, a), (e, é), (é, e), (i, í), (í, i), (o, ó),
(ó, o), (ö, ő), (ő, ö), (u, ú), (ú, u), (ü, ű), (ű, ü) }

Adjuk meg a típust!
$\mathcal{T} = (\varrho, I, \{S\})$, ahol $\varrho \subseteq E^* \times MGH$,

$\varrho =$ {(< 0 >, a), (< 14 >, á), (< 1 >, e), (< 13 >, é), (< 2 >, i),
(< 12 >, í), (< 3 >, o), (< 11 >, ó), (
(< 5 >, u), (< 9 >, ú), (< 6 >, ü), (<

$\forall \alpha \in E^*$:

$$I(\alpha) = (|\alpha| = 1 \wedge \alpha_1 \neq 7)$$

$S \subseteq E^* \times (E^*)^*$,

$$\begin{array}{rcl} S & = & \{(\langle i \rangle, \langle\langle i \rangle, \langle 14 - i \rangle\rangle) \mid i \in E\} \cup \\ & \cup & \{(\alpha, <\alpha, \alpha, \cdots>) \mid |\alpha| \neq 1\} \end{array}$$

Az, hogy a most megadott típus megfelel a fenti típusspecifikációnak, könnyen látható: a reprezentáció helyessége a ϱ és az I definíciójából leolvasható, míg az, hogy az S program a ϱ-n keresztül megoldja az F feladatot, a program egyszerű hozzárendeléséből és a ϱ „trükkös" megválasztásából látszik.

Természetesen másmilyen reprezentációs függvényt is meg lehet adni, de ekkor meg kell változtatnunk a típusinvarinánst és a programot is.

5.2. példa: Specifikáld azt a típust, melynek értékei a $[0..127]$ halmaz részhalmazai, típusműveletei pedig két részhalmaz metszetének ill. uniójának képzése, ill. annak megállapítása, hogy egy elem eleme-e egy részhalmaznak. Adj meg egy típust, amely megfelel a specifikációnak! (Az elemi értékek halmaza: $\{0, 1\}$, a programokat elég a programfüggvényükkel megadni.)
Megoldás: $\mathcal{T}_s = (T, I_s, \mathbb{F})$, ahol

$$\begin{array}{rcl} T & = & 2^{[0..127]}, \\ I_s & = & <igaz>, \\ \mathbb{F} & = & \{F_m, F_u, F_e\}, \end{array}$$

és $A_m = T \times T \times T, F_m \subseteq A_m \times A_m$,

$$F_m = \{((a,b,c),(p,q,r)) \mid p = a \wedge q = b \wedge r = a \cap b\}$$

$A_u = T \times T \times T, F_u \subseteq A_u \times A_u$

$$F_u = \{((a,b,c),(p,q,r)) \mid p = a \wedge q = b \wedge r = a \cup b\}$$

$A_e = T \times [0..127] \times \mathbb{L}, F_e \subseteq A_e \times A_e$

$$F_e = \{((h,e,l),(h',e',l')) \mid h = h' \wedge e = e' \wedge l' = (e \in h)\}$$

Adjunk a fenti specifikációnak megfelelő típust! $\mathcal{T} = (\varrho, I, \mathbb{S})$, és $\varrho \subseteq E^* \times 2^{\mathbb{N}}$, $\forall \alpha \in E^*$:

$$\varrho(\alpha) = \{\{i \mid \alpha_{i+1} = 1\}\},$$

$I : E^* \to \mathbb{L}, \forall \alpha \in E^*$:

$$I(\alpha) = (|\alpha| = 128),$$

$\mathbb{S} = \{S_m, S_u, S_e\}$, és $B_m = E^* \times E^* \times E^*,\ S_m \subseteq B_m \times B_m$ program

$$p(S_m) = \{((\alpha,\beta,\gamma),(\alpha',\beta',\gamma')) \quad | \quad I(\alpha) \wedge I(\beta) \wedge I(\gamma') \wedge \alpha = \alpha' \wedge \beta = \beta' \wedge \forall i \in [1..128] : \gamma'_i = \alpha_i * \beta_i\}$$

$B_u = E^* \times E^* \times E^*,\ S_u \subseteq B_u \times B_u$ program

$$p(S_u) = \{((\alpha,\beta,\gamma),(\alpha',\beta',\gamma')) \quad | \quad I(\alpha) \wedge I(\beta) \wedge I(\gamma') \wedge \alpha = \alpha' \wedge \beta = \beta' \wedge \forall i \in [1..128] : \gamma'_i = \alpha_i + \beta_i - \alpha_i * \beta_i\}$$

$B_e = E^* \times [0..127] \times \mathbb{L},\ S_e \subseteq B_e \times B_e$ program

$$p(S_e) = \{((\alpha,x,l),(\alpha',x',l')) \mid I(\alpha) \wedge \alpha = \alpha' \wedge x = x' \wedge l' = (\alpha_{x+1} = 1)\}$$

Vajon megfelel a most leírt típus a fenti típusspecifikációnak? A reprezentáció helyes, ugyanis a pontosan 128 hosszú sorozatokat a reprezentációs függvény éppen a kívánt részhalmazokba képezi le, azaz

$$\varrho([I]) = [I_s]$$

Vizsgáljuk meg a programok és a feladatok viszonyát. Vegyük észre, hogy a programok állapotterei illeszkednek a megfelelő feladat állapotteréhez, tehát felírható közöttük a γ reláció.

$$\begin{aligned} \gamma_m &= (\varrho\big|_{\lceil I \rceil}; \varrho\big|_{\lceil I \rceil}; \varrho\big|_{\lceil I \rceil}), \\ \gamma_u &= (\varrho\big|_{\lceil I \rceil}; \varrho\big|_{\lceil I \rceil}; \varrho\big|_{\lceil I \rceil}), \\ \gamma_e &= (\varrho\big|_{\lceil I \rceil}; id_{[0..127]}; id_{\mathbb{L}}) \end{aligned}$$

Ezek felhasználásával a ϱ-n keresztüli megoldás egyszerűen adódik a reprezentációs függvény és a programfüggvények szemantikájából.

5.3. példa: Legyen $\mathcal{T}_s = (T, I_s, \mathbb{F})$ egy típusspecifikáció, $\mathbb{F} = \{F\}$. Legyenek $\mathcal{T}_1 = (\varrho_1, I_1, \mathbb{S}_1)$ és $\mathcal{T}_2 = (\varrho_2, I_2, \mathbb{S}_2)$ típusok, melyekre: $\mathbb{S}_1 = \{S_1\}, \mathbb{S}_2 = \{S_2\}, \varrho_1 = \varrho_2, [I_1] = [I_2]$, és $S_2 \subseteq S_1$.
Igaz-e, hogy ha $\mathcal{T}_1$ megfelel $\mathcal{T}_s$-nek, akkor $\mathcal{T}_2$ is?
Megoldás: A reprezentáció helyessége $\varrho_1 = \varrho_2$ és $[I_1] = [I_2]$ miatt triviálisan teljesül, hiszen ekkor:

$$\varrho_2([I_2]) = \varrho_1([I_1]) = [I_s].$$

Azt kell tehát megvizsgálnunk, hogy vajon az S_2 program megoldja-e az F feladatot a ϱ_2-n keresztül. Mivel a programok állapottere közös, feltehetjük, hogy a programok állapottere és a feladat állapottere egymásnak megfeleltethető, hiszen ellenkező esetben mindkét megoldás-vizsgálatnál a feladatnak ugyanazt a kiterjesztését kellene használnunk, és így az eredeti feladatot ezzel a kiterjesztéssel helyettesítve az alábbi gondolatmenet végigvihető.

Mivel $S_2 \subseteq S_1$, a két program programfüggvényére teljesül a következő:

i. $\mathcal{D}_{p(S_1)} \subseteq \mathcal{D}_{p(S_2)}$,

ii. $\forall a \in \mathcal{D}_{p(S_1)} : p(S_2)(a) \subseteq p(S_1)(a)$.

Jelöljük most is γ-val a program és a feladat állapottere közötti, a megfeleltetésben definiált leképezést. Könnyen látható, hogy az $i.$ tulajdonság miatt

$$\mathcal{D}_{\gamma\odot p(S_1)\odot\gamma^{(-1)}} \subseteq \mathcal{D}_{\gamma\odot p(S_2)\odot\gamma^{(-1)}}.$$

Másrészt mivel az S_1 program megoldja F-et a ϱ-n keresztül

$$\mathcal{D}_F \subseteq \mathcal{D}_{\gamma\odot p(S_1)\odot\gamma^{(-1)}}$$

is teljesül. A fenti két állítás alapján

$$\mathcal{D}_F \subseteq \mathcal{D}_{\gamma\odot p(S_2)\odot\gamma^{(-1)}}.$$

Használjuk fel a második tulajdonságot is! Az $ii.$ tulajdonság miatt igaz az alábbi állítás is:

$$\forall a \in \mathcal{D}_{\gamma\odot p(S_1)\odot\gamma^{(-1)}} : \gamma \odot p(S_2) \odot \gamma^{(-1)}(a) \subseteq \gamma \odot p(S_1) \odot \gamma^{(-1)}(a).$$

Ekkor viszont mivel az S_1 program – a ϱ-n keresztül – megoldása a feladatnak, teljesül, hogy

$$\forall a \in \mathcal{D}_F : \gamma \odot p(S_1) \odot \gamma^{(-1)}(a) \subseteq F(a),$$

és ezért

$$\forall a \in \mathcal{D}_F : \gamma \odot p(S_2) \odot \gamma^{(-1)}(a) \subseteq F(a),$$

azaz az S_2 program is megoldja az F feladatot a ϱ-n keresztül, tehát a $\mathcal{T}_2$ típus is megfelel a specifikációnak.

5.5. Feladatok

1. Adjunk típusspecifikációt, reprezentációs függvényt, típust (ami megfelel a specifikációnak) a következő típusra: a lehetséges értékek: $[0..99999]$. A műveletek a következő és az előző 100000 szerinti maradékkal. Az elemi értékek a decimális számjegyek $\{0,1,2,3,4,5,6,7,8,9\}$. Mutasd meg, hogy a típus megfelel a típusspecifikációnak!

2. $E = \{0,1,2\}, T_s = \{1,2,3,4,5,6,7,8,9,0\}, \mathbb{F} = \{F\}$.

   $$F = \{((a,b,c),(d,e,f)) \mid \exists k \in \mathbb{Z} : f + k * 10 = a + b\}$$

   Készíts el egy olyan típust, ami megfelel a specifikációnak!

3. Legyen $\mathcal{T}_{s1} = (T, I_{s1}, \mathbb{F}_1), \mathcal{T}_{s2} = (T, I_{s2}, \mathbb{F}_2)$ két típusspecifikáció!

   1. állítás: Minden $\mathcal{T}$ típusra: $\mathcal{T}$ megfelel $\mathcal{T}_{s1}$-nek $\Leftrightarrow$ $\mathcal{T}$ megfelel $\mathcal{T}_{s2}$-nek.
   2. állítás: $[I_{s1}] = [I_{s2}]$ és $\mathbb{F}_1 = \mathbb{F}_2$.

   Ekvivalens-e a két állítás?

4. Adott a $\mathcal{T}_s = (T, I_s, \mathbb{F})$ típusspecifikáció, továbbá adottak a $\mathcal{T}_1 = (\varrho_1, I_1, \mathbb{S}_1)$, $\mathcal{T}_2 = (\varrho_2, I_2, \mathbb{S}_2)$ típusok. Tegyük fel, hogy $[I_1] = [I_2], \mathbb{S}_1 = \mathbb{S}_2$ és $\varrho_1([I_1]) = \varrho_2([I_2])$ és $\forall \alpha \in E^* : \ \varrho_2(\alpha) \subseteq \varrho_1(\alpha)$, valamint $\mathcal{T}_1$ feleljen meg $\mathcal{T}_s$-nek!

   Igaz-e, hogy $\mathcal{T}_2$ is megfelel $\mathcal{T}_s$-nek?

5. Legyen $\mathcal{T}_s = (T, I_s, \mathbb{F})$ egy típusspecifikáció, $\mathcal{T}_1 = (\varrho_1, I_1, \mathbb{S}_1), \mathcal{T}_2 = (\varrho_2, I_2, \mathbb{S}_2)$. Legyen $[I_2] \subseteq [I_1], \mathbb{S}_1 = \mathbb{S}_2$ és $\varrho_1([I_1]) = \varrho_2([I_2])$, és $\forall \alpha \in E^* : \ \varrho_2(\alpha) = \varrho_1(\alpha)$, valamint $\mathcal{T}_1$ feleljen meg $\mathcal{T}_s$-nek!

   Igaz-e, hogy $\mathcal{T}_2$ is megfelel $\mathcal{T}_s$-nek?

6. fejezet

Elemi programok

Ebben a fejezetben bevezetünk néhány egyszerű programot. Ezeket a programokat a következő fejezetekben gyakorta fogjuk használni, ezért megvizsgáljuk a program-függvényüket, és egy adott utófeltételhez tartozó leggyengébb előfeltételüket.

6.1. Definíció (ELEMI PROGRAM). *Eleminek nevezünk egy A állapottéren egy S programot, ha*

$$\forall a \in A: \ S(a) \subseteq \{< a >, < a, a, a, \cdots >, < a, b > \ | \ b \neq a\}.$$

Az elemi programok közül is kiválasztunk néhány speciális tulajdonsággal rendelkezőt, és a továbbiakban csak velük foglalkozunk.

Az első ilyen program, az üres program lesz, ami nem csinál semmit.

6.2. Definíció (SKIP). *SKIP-nek nevezzük azt a programot, amire*

$$\forall a \in A: \ SKIP(a) = \{< a >\}.$$

A második program a törlődés, aminek legfontosabb jellemzője, hogy soha sem terminál.

6.3. Definíció (ABORT). *ABORT-tal jelöljük azt a programot, amire*

$$\forall a \in A: \ ABORT(a) = \{< a, a, a, \cdots >\}.$$

Az eddig felsorolt két elemi program segítségével még bajosan tudnánk egy adott feladatot megoldó programot készíteni, hiszen egyrészt nem túl sok olyan feladat van, aminek az $ABORT$ program megoldása lenne (vajon van ilyen egyáltalán?) és a $SKIP$ program is csak egy nagyon szűk feladatosztálynak lehet megoldása (melyek ezek a feladatok?). A harmadik – és a programozási feladat megoldása szempontjából legfontosabb – speciális elemi program az értékadás.

6.4. Definíció (ÉRTÉKADÁS). *Legyen $A = A_1 \times \cdots \times A_n$, $F = (F_1, \ldots, F_n)$, ahol $F_i \subseteq A \times A_i$. Az S program általános értékadás, ha*

$$\begin{aligned} S \quad = \quad & \{(a, red(< a, b >)) \mid a, b \in A \wedge a \in \bigcap_{i=1}^{n} \mathcal{D}_{F_i} \wedge b \in F(a)\} \cup \\ & \{(a, < a, a, a, \cdots >) \mid a \in A \wedge a \notin \bigcap_{i=1}^{n} \mathcal{D}_{F_i}\} \end{aligned}$$

A defi níció alapján könnyen látható, hogy az $F \subseteq A \times A$ relációra fennáll az alábbi tulajdonság:

- $\mathcal{D}_F = \bigcap_{i=1}^{n} \mathcal{D}_{F_i}$,
- $F(a) = F_1(a) \times F_2(a) \times \cdots \times F_n(a)$

A fenti F_i komponensrelációk tehát pontosan azt írják le, hogy az adott értékadás miként változtatja meg az állapottér egyes komponenseit.

6.5. Definíció (AZ ÁLTALÁNOS ÉRTÉKADÁS SPECIÁLIS ESETEI).

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

Ha egy kivételével az összes F_i projekció – azaz az értékadás az állapottérnek csak egy komponensét (csak egy változó értékét) változtatja meg –, akkor S-et *egyszerű* értékadásnak, egyébként *szimultán* értékadásnak nevezzük.

Az értékadás egy kicsit bonyolultabb mint előző két társa, de egy kicsit „értékesebb" is, hiszen értékadással minden feladat megoldható! A kérdés persze csupán az, hogy az éppen adott feladat által definiált értékadás megengedett művelet-e. Ezzel a kérdéssel a későbbiekben – a programozási feladat megoldása során – fogunk foglalkozni.

Vizsgáljuk meg a fent definiált speciális elemi programok programfüggvényeit!

6.5. TÉTEL: ELEMI PROGRAMOK PROGRAMFÜGGVÉNYE

1. $p(SKIP) = id_A$,
2. $p(ABORT) = \emptyset$,
3. $p(a := F(a)) = F$,
4. $p(a :\in F(a)) = F$.

A tételt bizonyítása triviális, ezért itt nem bizonyítjuk (a feladatok között szerepel).

Most, hogy megvizsgáltuk a programfüggvényeket, nézzük meg az elemi programok adott utófeltételhez tartozó leggyengébb előfeltételét.

Mivel a SKIP program programfüggvénye az identikus leképezés, egy tetszőleges R utófeltételhez tartozó leggyengébb előfeltétele:

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

Hasonlóan egyszerűen látható, hogy – mivel az $ABORT$ program programfüggvénye üres – a leggyengébb előfeltétel definíciója alapján egy tetszőleges R utófeltétel esetén

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

Az általános értékadás leggyengébb előfeltételét külön vizsgáljuk a determinisztikus és nem determinisztikus, illetve a globális, és a parciális esetben.

Legyen $F : A \to A$ függvény. Ekkor

$$\begin{aligned} \lceil lf(a := F(a), R)\rceil &= \{a \in A \mid F(a) \in \lceil R\rceil\} = \\ &= F^{(-1)}(\lceil R\rceil) = F^{-1}(\lceil R\rceil) = \lceil R \circ F\rceil. \end{aligned}$$

Ha F parciális függvény, akkor az általa definiált értékadás is parciális, melynek leggyengébb előfeltétele:

$$\begin{aligned} \lceil lf(a := F(a), R)\rceil &= \{a \in A \mid F(a) \in \lceil R\rceil\} \cap \mathcal{D}_F = \\ &= F^{(-1)}(\lceil R\rceil) \cap \mathcal{D}_F = F^{-1}(\lceil R\rceil) \cap \mathcal{D}_F. \end{aligned}$$

Most vizsgáljuk azt a két esetet, amikor F nem determinisztikus. Feltéve, hogy $\mathcal{D}_F = A$, az értékkiválasztás leggyengébb előfeltétele

$$\lceil lf(a :\in F(a), R)\rceil = \{a \in A \mid F(a) \subseteq \lceil R\rceil\} = F^{-1}(\lceil R\rceil).$$

Ugyanez parciális esetben:

$$\lceil lf(a :\in F(a), R)\rceil = \{a \in A \mid F(a) \subseteq \lceil R\rceil\} \cap \mathcal{D}_F = F^{-1}(\lceil R\rceil) \cap \mathcal{D}_F.$$

Az értékadást általában változókkal írjuk le. Legyenek az állapottér változói rendre $x_1, x_2, \ldots, x_n$. Ekkor az $a := F(a)$ program jelölésére az alábbi formulát használhatjuk.

$$x_1, x_2, \ldots, x_n := F_1(x_1, x_2, \ldots, x_n), F_2(x_1, x_2, \ldots, x_n), \ldots, F_n(x_1, x_2, \ldots, x_n).$$

A gyakorlatban az esetek többségében F komponenseinek nagy része projekció, azaz $F_i(x_1, x_2, \ldots, x_n) = x_i$. Ekkor az értékadás jelöléséből a bal oldalról x_i-t , a jobb oldalról pedig $F_i(x_1, x_2, \ldots, x_n)$-t elhagyjuk. Vegyük észre, hogy egyszerű értékadás esetén a bal oldalon csak egy változó, a jobb oldalon pedig csak egy kifejezés marad.

Jelölésünket abban az esetben még tovább egyszerűsíthetjük, ha az értékadás jobb oldala (F_i) nem függ minden változótól. Ekkor a jobb oldalon csak azokat a változókat tüntetjük fel, amelyektől F_i függ.

Nézzük meg egy egyszerű példán a fent leírtakat: Legyen az állapottér

$$\begin{array}{ccc} A = & \mathbb{Z} & \times \mathbb{L} \\ & x & l \end{array}$$

A továbbiakban a fentihez hasonlóan az állapottér egyes komponenseihez tartozó változókat a komponensek alá fogjuk írni. Legyenek az értékadás komponensei: $\forall a = (a_1, a_2) \in A$:

$$\begin{aligned} F_1(a_1, a_2) &= a_1, \text{ azaz } F_1 = pr_{\mathbb{Z}}, \text{ és} \\ F_2(a_1, a_2) &= (a_1 > 0). \end{aligned}$$

Ekkor az $a := F(a)$ értékadás változókkal felírva:

$$x, l := x, (x > 0).$$

A jelölés fent leírt egyszerűsítéseit elvégezve az

$$l := (x > 0)$$

egyszerű értékadást kapjuk.

Ha felhasználjuk, hogy az értékadás leggyengébb előfeltétele $R \circ F$, akkor egy változókkal felírt értékadás változókkal megadott előfeltételét egyszerűen kiszámíthatjuk: helyettesítsük az utófeltételben az értékadásban szereplő változókat az új értékükkel. Erre bevezetünk egy új jelölést is: legyenek $x_1, x_2, \ldots, x_n$ rendre az állapottér változói. Ekkor

$$lf(x_{i_1}, \ldots, x_{i_m} := F_{i_1}(x_1, \ldots, x_n), \ldots, F_{i_m}(x_1, \ldots, x_n), R) = R^{x_{i_1} \leftarrow F_1(x_1,\ldots,x_n),\ldots,x_{i_m} \leftarrow F_m(x_1,\ldots,x_n)}$$

6.1. Feladatok

1. $A = \{1, 2, 3\}, B = \{a, b\}, C = A \times B$. Legyen S program A-n, $S = \{1 \to \langle 1 \rangle, 2 \to \langle 2222 \ldots \rangle, 3 \to \langle 31 \rangle\}$.

   Legyen S_1 az S kiterjesztése C-re, M pedig olyan program C-n, hogy M ekvivalens S-sel A-n.

   (a) elemi program-e S?

   (b) elemi program-e S_1 és biztosan elemi program-e M?

2. Tekintsük az alábbi állapotteret:

   $$A = \begin{matrix} \mathbb{N} & \times & \mathbb{N} \\ x & & y \end{matrix}$$

   Mi az $(x, y) := F(x, y),\ F = (F_1, F_2),\ F_1(x, y) = y,\ F_2(x, y) = x$, azaz az $F(p, q) = \{b \in A \mid x(b) = q \wedge y(b) = p\}$ értékadás $R = (x < y)$ utófeltételhez tartozó leggyengébb előfeltétele?

3. Legyen A tetszőleges állapottér. Melyek azok a feladatok az A-n, amelyeknek megoldása a $SKIP$ program?

4. Legyen A tetszőleges állapottér. Melyek azok a feladatok az A-n, amelyeknek megoldása a $ABORT$ program?

7. fejezet

Programkonstrukciók

Ebben a fejezetben azzal foglakozunk, hogy hogyan lehet meglevő programokból új programokat készíteni. Természetesen sokféleképpen konstruálhatunk meglevő programjainkból új programokat, de most csak háromféle konstrukciós műveletet fogunk megengedni: a szekvencia-, elágazás- és ciklusképzést. Később megmutatjuk, hogy ez a három konstrukciós művelet elegendő is. Nemcsak definiáljuk ezeket a konsrukciókat, de megvizsgáljuk programfüggvényüket és leggyengébb előfeltételüket is.

7.1. Megengedett konstrukciók

Az első definíció arról szól, hogy egy programot közvetlenül egy másik után végezzünk el. A definícióban használni fogjuk a következő jelölést: legyen $\alpha \in A^*$, $\beta \in A^{**}$,

$$\chi_2(\alpha, \beta) ::= red(kon(\alpha, \beta)).$$

7.1. Definíció (SZEKVENCIA). *Legyenek $S_1, S_2 \subseteq A \times A^{**}$ programok. Az $S \subseteq A \times A^{**}$ relációt az S_1 és S_2* szekvenciájának *nevezzük, és $(S_1; S_2)$-vel jelöljük, ha $\forall a \in A$:*

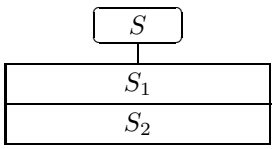
$$\begin{aligned} S(a) &= \{\alpha \in A^\infty \mid \alpha \in S_1(a)\} \cup \\ &\cup \{\chi_2(\alpha, \beta) \in A^{**} \mid \alpha \in S_1(a) \cap A^* \wedge \beta \in S_2(\tau(\alpha))\} \end{aligned}$$

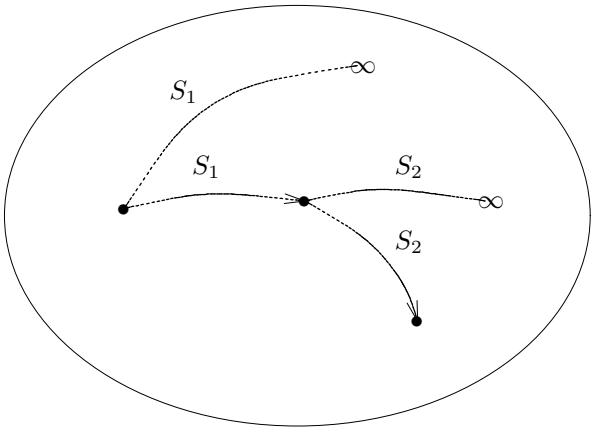
Vegyük észre, hogy ha két olyan program szekvenciáját képezzük, amelyek értékkészlete csak véges sorozatokat tartalmaz, akkor a szekvencia is csak véges sorozatokat rendel az állapottér pontjaihoz.

Hasonlóan egyszerűen ellenőrizhető az is, hogy determinisztikus programok szekvenciája is determinisztikus reláció (függvény).

A programkonstrukciókat szerkezeti ábrákkal, úgynevezett **struktogramokkal** szoktuk ábrázolni.

Legyenek S_1, S_2 programok A-n. Ekkor az $S = (S_1; S_2)$ szekvencia struktogramja:





7.1. ábra. Szekvencia

7.2 A második konstrukciós lehetőségünk az, hogy más-más meglevő programot hajtsunk végre bizonyos feltételektől függően.

7.2. Definíció (ELÁGAZÁS).
Legyenek $\pi_1, \ldots, \pi_n : A \to \mathbb{L}$ feltételek, $S_1, \ldots, S_n$ programok A-n. Ekkor az $IF \subseteq A \times A^{**}$ relációt az S_i-kből képzett π_i-k által meghatározott *elágazásnak* nevezzük és $(\pi_1 : S_1, \ldots, \pi_n : S_n)$-nel jelöljük, ha $\forall a \in A$:

$$IF(a) = \bigcup_{i=1}^{n} w_i(a) \cup w_0(a),$$

ahol $\forall i \in [1..n]$:

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

és

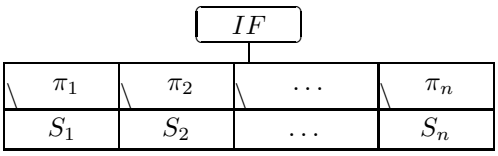
$$w_0(a) \quad = \quad \begin{cases} \langle a, a, a, \ldots \rangle, & \text{ha } \forall i \in [1..n] : \neg\pi_i(a) \\ \emptyset, & \text{különben} \end{cases}$$

Az elágazás definíciójában nem kötöttük ki, hogy a feltételek diszjunktak, tehát az állapottér egy pontjában több feltétel is igaz lehet, ebben az esetben az elágazás az összes olyan sorozatot hozzárendeli ehhez a ponthoz, amit legalább egy olyan program, aminek a feltételrésze ebben a pontban igaz. Ezért haa a feltételek nem diszjunktak, akkor a determinisztikus programokból képzett elágazás lehet nem determinisztikus is.

Ha csak olyan programokból képzünk elágazást, amik csak véges sorozatokat rendelnek az állapottér minden pontjához, az elágazás akkor is rendelhet (azonos elemekből álló) végtelen sorozatot, ha a feltételek igazsághalmazai nem fedik le az egész állapotteret.

Megjegyezzük, hogy a definíció szerint, ha egy pontban egyik feltétel sem teljesül, az elágazás "elszáll", ellentétben a legtöbb gyakorlatilag használt programnyelvvel.

Legyenek $S_1, S_2, \ldots, S_n$ programok, $\pi_1, \pi_2, \ldots, \pi_n$ feltételek A-n. Ekkor az $IF = (\pi_1 : S_1, \pi_2 : S_2, \ldots, \pi_n : S_n)$ elágazás struktogramja:



A harmadik konstrukciós lehetőségünk az, hogy egy meglevő programot egy feltételtől függően valahányszor egymás után végrehajtsunk. A ciklus definiálásához szükségünk van további két jelölésre: véges sok, illetve végtelen sok sorozat konkatenációjának redukáltjára.

Legyenek $\alpha^1, \alpha^2, \ldots, \alpha^{n-1} \in A^*$ és $\alpha^n \in A^{**}$,

$$\chi_n(\alpha^1, \alpha^2, \ldots, \alpha^n) ::= red(kon(\alpha^1, \alpha^2, \ldots, \alpha^n)).$$

Legyenek $\alpha^i \in A^*$ $(i \in \mathbb{N})$,

$$\chi_\infty(\alpha^1, \alpha^2, \ldots) ::= red(kon(\alpha^1, \alpha^2, \ldots)).$$

7.3. Definíció (CIKLUS). *Legyen π feltétel és S_0 program A-n. A $DO \subseteq A \times A^{**}$ relációt az S_0-ból a π feltétellel képzett ciklusnak nevezzük, és (π, S_0)-lal jelöljük, ha*

- $\forall a \notin \lceil \pi \rceil$:
$$DO(a) = \{\langle a \rangle\}$$
- $\forall a \in \lceil \pi \rceil$:

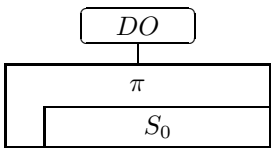
$$\begin{aligned} DO(a) \quad = \quad & \{\alpha \in A^{**} \mid \exists \alpha^1, \ldots, \alpha^n \in A^{**} : \alpha = \chi_n(\alpha^1, \ldots, \alpha^n) \wedge \\ & \alpha^1 \in S_0(a) \wedge \forall i \in [1..n-1] : \ \alpha^i \in A^* \wedge \\ & \alpha^{i+1} \in S_0(\tau(\alpha^i)) \wedge \pi(\tau(\alpha^i)) \wedge (\alpha^n \in A^\infty \vee \\ & (\alpha^n \in A^* \wedge \neg\pi(\tau(\alpha^n))))\} \ \cup \\ \cup \quad & \{\alpha \in A^\infty \mid \forall i \in \mathbb{N} : \exists \alpha^i \in A^* : \alpha = \chi_\infty(\alpha^1, \alpha^2, \ldots) \wedge \\ & \alpha^1 \in S_0(a) \ \wedge \forall i \in \mathbb{N} : \ \alpha^i \in A^* \wedge \alpha^{i+1} \in S_0(\tau(\alpha^i)) \wedge \\ & \pi(\tau(\alpha^i))\}. \end{aligned}$$

Első ránézésre a definíció kissé bonyolult. Nézzük meg alaposabban!

A definícó alapján nyílvánvaló, hogy a determinisztikus programból képzett ciklus is determinisztikus lesz.

Ezzel ellentétben, ha egy csak véges sorozatokat rendelő programot foglalunk ciklusba, akkor a ciklus értékkészlete még tartalmazhat végtelen sorozatot (ha soha nem jutunk ki a feltétel igazsághalmazából).

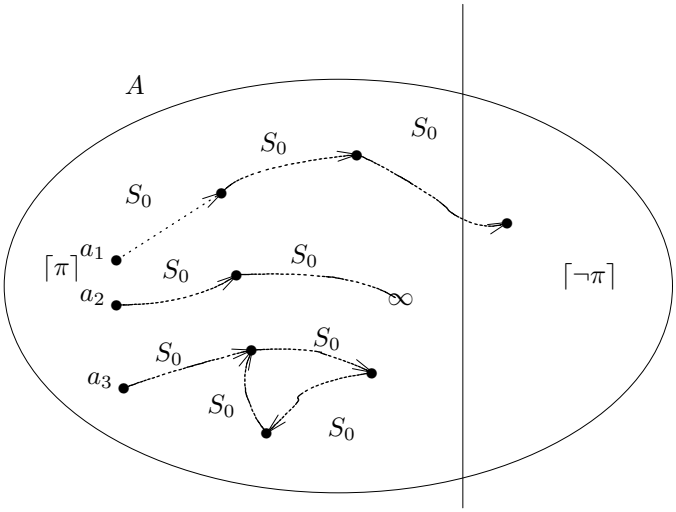
Legyen S_0 program, π feltétel A-n. Ekkor a $DO = (\pi, S_0)$ ciklus struktogramja:



A fentiekben leírt konstrukciókat programokra lehet alkalmazni. De vajon programokat hoznak-e létre? Az alábbi tétel kimondja, hogy az előzőkben definiált három konstrukciós művelet meglevő programokból valóban programokat hoz létre.

7.6. TÉTEL: A SZEKVENCIA, AZ ELÁGAZÁS ÉS A CIKLUS PROGRAM.

Legyen A tetszőleges állapottér, $S_0, S_1, S_2, \ldots, S_n \subseteq A \times A^{**}$ programok, valamint $\pi, \pi_1, \pi_2, \ldots, \pi_n : A \to \mathbb{L}$ feltételek A-n. Ekkor



7.2. ábra. Ciklus

1. $S = (S_1; S_2)$,
2. $IF = (\pi_1 : S_1, \pi_2 : S_2, \ldots, \pi_n : S_n)$,
3. $DO = (\pi, S_0)$

programok A-n.

Bizonyítás: Mindhárom konstrukció definíciójában explicit szerepel, hogy reláció $A \times A^{**}$-on, és $\mathcal{D}_S = A$, azaz teljesül a program definíciójának első pontja. A továbbiakban esetekre bontva megvizsgáljuk a másik két kritérium teljesülését.

1. Legyen $a \in A$ tetszőleges, és $\alpha \in S(a)$. Ekkor két eset lehetséges:
   - Ha $\alpha \in S_1(a)$ és ebben az esetben $\alpha_1 = a$ és $\alpha = red(\alpha)$ triviálisan teljesül, hiszen S_1 program.
   - Ha $\alpha = \chi_2(\alpha^1, \alpha^2)$ úgy, hogy $\alpha^1 \in S_1(a)$ és $\alpha^2 \in S_2(\tau(\alpha^1))$. Ekkor a χ_2 definíciója miatt α redukált. Másrészt $\alpha_1 = \alpha^1_1$, tehát $\alpha_1 = a$.
2. Legyen $a \in A$ tetszőleges, és $\alpha \in IF(a)$. Ekkor
$$\alpha \in \bigcup_{i=0}^{n} w_i(a).$$
   - Tegyük fel, hogy $\alpha \in w_0(a)$. Ekkor $\alpha = \langle a, a, \ldots \rangle$, tehát teljesíti a program definíciójának kritériumait.
   - Tegyük fel, hogy $\exists i \in [1..n] : \alpha \in w_i(a)$. Ekkor $\alpha \in S_i(a)$, így mivel S_i program, α teljesíti a definícióban megkívánt tulajdonságokat.
3. Legyen $a \in A$ tetszőleges, és $\alpha \in DO(a)$.
   - Tegyük fel, hogy $\neg\pi(a)$. Ekkor $\alpha = \langle a \rangle$, így az előírt tulajdonságok triviálisan teljesülnek.
   - Tegyük fel, hogy $\pi(a)$. Ekkor három eset lehetséges:

(a) $\alpha \in A^*$:
Ekkor $\exists n \in \mathbb{N} : \alpha = \chi_n(\alpha^1, \ldots, \alpha^n)$, így – χ_n definíciója miatt – α redukált. Másrészt felhasználva, hogy $\alpha^1 \in S_0(a)$ és $\alpha_1 = \alpha_1^1$, $\alpha_1 = a$ is teljesül.

(b) $\exists n \in \mathbb{N} : \alpha = \chi_n(\alpha^1, \ldots, \alpha^n) \wedge \forall i \in [1..n-1] : \alpha^i \in A^* \wedge \alpha_n \in A^\infty$: Ekkor a kritériumok teljesülése az előző ponttal analóg módon ellenőrizhető.

(c) $\alpha = \chi_\infty(\alpha^1, \alpha^2, \ldots)$:
Ekkor α a χ_∞ definíciója alapján redukált sorozat, és $\alpha_1 = \alpha_1^1 = a$ is teljesül.

□

7.2. A programkonstrukciók programfüggvénye

Miután beláttuk, hogy meglevő programokból a programkonstrukciók segítségével új programokat készíthetünk, vizsgáljuk meg, hogy milyen kapcsolat van a konstruált programok programfüggvénye és az eredeti programok programfüggvénye között.

A szekvencia a legegyszerűbb programkonstrukció, ennek megfelelően a programfüggvénye is egyszerűen felírható a két komponensprogram programfüggvényének segítségével. Mivel a szekvencia két program egymás utáni elvégzését jelenti, várható, hogy a programfüggvénye a két komponensprogram programfüggvényének kompozíciója. Azonban mint látni fogjuk kompozíció helyett szigorú kompozíciót kell alkalmazni.

7.7. TÉTEL: A SZEKVENCIA PROGRAMFÜGGVÉNYE

Legyen A tetszőleges állapottér, S_1, S_2 programok A-n, $S = (S_1; S_2)$. Ekkor

$$p(S) = p(S_2) \odot p(S_1).$$

Bizonyítás: Legyen $a \in A$ tetszőleges. Ekkor

$$\begin{gathered} a \in \mathcal{D}_{p(S)} \Longleftrightarrow S(a) \subseteq A^* \Longleftrightarrow \\ S_1(a) \subseteq A^* \wedge \forall \alpha \in S_1(a) : S_2(\tau(\alpha)) \subseteq A^* \Longleftrightarrow \\ a \in \mathcal{D}_{p(S_1)} \wedge p(S_1)(a) \subseteq \mathcal{D}_{p(S_2)} \Longleftrightarrow \\ a \in \mathcal{D}_{p(S_2) \odot p(S_1)} \end{gathered}$$

tehát:

$$\mathcal{D}_{p(S)} = \mathcal{D}_{p(S_2) \odot p(S_1)}.$$

Legyen $a \in \mathcal{D}_{p(S)}$. Ekkor

$$\begin{gathered} (a, a') \in p(S_2) \odot p(S_1) \Longleftrightarrow \\ \exists b \in A : (a, b) \in p(S_1) \wedge (b, a') \in p(S_2) \Longleftrightarrow \\ \exists \alpha \in S_1(a), \beta \in S_2(b) : \tau(\alpha) = b \wedge \tau(\beta) = a' \Longleftrightarrow \\ (a, a') \in p(S). \end{gathered}$$

□

Vegyük észre, hogy a nem szigorú kompozíció értelmezési tartományában olyan pont is lehet, amelyhez a szekvencia rendel végtelen sorozatot is. Nézzünk erre egy egyszerű példát: Legyen $A = \{1, 2\}$,

$$\begin{aligned} S_1 &= \{(1, \langle 1 \rangle), (1, \langle 1, 2 \rangle), (2, \langle 2 \rangle)\}, \\ S_2 &= \{(1, \langle 1, 2 \rangle), (2, \langle 2, 2, \ldots \rangle)\}. \end{aligned}$$

Ekkor $1 \in \mathcal{D}_{p(S_2)\circ p(S_1)}$, de $\langle 1,2,2,2,\dots\rangle \in S(1)$.

Mivel az elágazást több programból képezzük, a programfüggvényét is csak kissé körülményesebben tudjuk megfogalmazni. Hiszen az, hogy egy ponthoz az elágazás rendel-e végtelen sorozatot attól is függ, hogy mely feltételek igazak az adott pontban. Sőt, ha egy pontban egyetlen feltétel sem igaz, akkor a komponensprogramok programfüggvényétől függetlenül abban a pontban az elágazás programfüggvénye nem lesz értelmezve. Az elágazás programfüggvényének formális megfogalmazását adja meg a következő tétel.

7.8. TÉTEL: AZ ELÁGAZÁS PROGRAMFÜGGVÉNYE

Legyen A tetszőleges állapottér, $S_1, S_2, \dots, S_n \subseteq A \times A^{**}$ programok, valamint $\pi_1, \pi_2, \dots, \pi_n : A \to \mathbb{L}$ feltételek A-n, $IF = (\pi_1 : S_1, \dots, \pi_n : S_n)$. Ekkor

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

$$p(IF)(a) = \bigcup_{i=1}^{n} pw_i(a),$$

ahol

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

Bizonyítás: Legyen $a \in A$ tetszőleges. Ekkor

$$\begin{gathered} a \in \mathcal{D}_{p(IF)} \Longleftrightarrow IF(a) \subseteq A^* \Longleftrightarrow \\ \exists i \in [1..n] : \pi_i(a) \wedge \bigcup_{i=1}^{n} w_i(a) \subseteq A^* \Longleftrightarrow \\ \exists i \in [1..n] : \pi_i(a) \wedge \forall i \in [1..n] : \pi_i(a) \Rightarrow a \in \mathcal{D}_{p(S_i)}. \end{gathered}$$

Legyen továbbá $a \in \mathcal{D}_{p(IF)}$. Ekkor

$$p(IF)(a) = \tau\left(\bigcup_{i=1}^{n} w_i(a)\right) = \bigcup_{i=1}^{n} pw_i(a).$$

□

Természetesen, ha az elágazás-feltételek lefedik az egész állapotteret, akkor az a feltétel, hogy valamelyik π_i-nek igaznak kell lennie, triviálisan teljesül.

Ahhoz, hogy a ciklus programfüggvényét egyszerűen megfogalmazhassuk, bevezetünk néhány további fogalmat.

7.9. TÉTEL: A CIKLUS PROGRAMFÜGGVÉNYE

Legyen A tetszőleges állapottér, S program, π feltétel A-n, $DO = (\pi, S)$. Ekkor

$$p(DO) = \overline{p(S)|_\pi}.$$

Bizonyítás: Legyen $a \in A$ tetszőleges. Ekkor

$$a \in \mathcal{D}_{p(DO)} \Longleftrightarrow DO(a) \subseteq A^*$$
$$\Updownarrow$$
$$DO(a) = \begin{cases} \{\langle a \rangle\}, & \text{ha } \neg\pi(a) \\ \{\alpha \in A^* \mid \exists n \in \mathbb{N} : \alpha = \chi_n(\alpha^1, \ldots, \alpha^n) \wedge \\ \quad \alpha^1 \in S(a) \wedge \forall i \in [1..n-1] : \\ \quad \alpha^{i+1} \in S(\tau(\alpha^i)) \wedge \pi(\tau(\alpha^i)) \wedge \\ \quad \neg\pi(\tau(\alpha^n))\}, & \text{ha } \pi(a) \end{cases}$$
$$\Updownarrow$$
$$\neg\pi(a) \vee (\nexists \beta \in A^\infty : \beta_1 = a \wedge \forall i \in \mathbb{N} : \beta_{i+1} \in p(S)(\beta_i) \wedge \pi(\beta_i))$$
$$\Updownarrow$$
$$a \in \mathcal{D}_{\overline{p(S)|_\pi}}$$

tehát $\mathcal{D}_{p(DO)} = \mathcal{D}_{\overline{p(S)|_\pi}}$. Másrészt legyen $a \in \mathcal{D}_{p(DO)}$. Ekkor

- Ha $\neg\pi(a)$ akkor
$$p(DO)(a) = a = \overline{p(S)|_\pi}(a).$$
- Ha $\pi(a)$ akkor $p(DO)(a) =$

$$\begin{aligned} &= \tau(\{\alpha \in A^* \mid \exists n \in \mathbb{N} : \alpha = \chi_n(\alpha^1, \ldots, \alpha^n) \wedge \alpha^1 \in S(a) \wedge \\ &\qquad \forall i \in [1..n-1] : \alpha^{i+1} \in S(\tau(\alpha^i)) \wedge \pi(\tau(\alpha^i)) \wedge \neg\pi(\tau(\alpha^n))\}) = \\ &= \tau(\{\beta \in A^* \mid \beta_1 = a \wedge \forall i \in [1..|\beta|-1] : \beta_{i+1} \in p(S)(\beta_i) \wedge \\ &\qquad \pi(\beta_i) \wedge \neg\pi(\tau(\beta))\}) = \\ &= \{b \in A \mid \exists k \in \mathbb{N} : b \in (p(S)|_\pi)^k(a) \wedge \neg\pi(b)\} \\ &= \overline{p(S)|_\pi}(a). \end{aligned}$$

□

7.3. Levezetési szabályok

Ebben az alfejezetben megvizsgáljuk a programkonstrukciók és a specifikáció kapcsolatát.

Először azt fogjuk megvizsgálni, hogy a szekvencia adott utófeltételhez tartozó leggyengébb előfeltétele milyen kapcsolatban van az őt alkotó programok leggyengébb előfeltételével.

7.10. TÉTEL: A SZEKVENCIA LEVEZETÉSI SZABÁLYA

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 $Q \Rightarrow lf(S, R)$.

Bizonyítás: Legyen $q \in \lceil Q \rceil$. Ekkor (1) miatt $q \in \mathcal{D}_{p(S_1)}$ és $p(S_1)(q) \subseteq \lceil Q' \rceil$. Mivel (2) miatt $\lceil Q' \rceil \subseteq \mathcal{D}_{p(S_2)}$: $q \in \mathcal{D}_{p(S_2) \odot p(S_1)} = \mathcal{D}_{p(S)}$.
Továbbá (2) miatt $p(S_2)\big(p(S_1)(q)\big) \subseteq \lceil R \rceil$, tehát $q \in lf(S, R)$.
□

A szekvencia levezetési szabálya és a specifikáció tétele alapján a következőt mondhatjuk: ha S_1 és S_2 olyan programok, amelyekre a paramétertér minden b pontjában $Q_b \Rightarrow lf(S_1, Q'_b)$ és $Q'_b \Rightarrow lf(S_2, R_b)$ teljesül, akkor $(S_1; S_2)$ megoldja a Q_b, R_b párokkal adott feladatot.

7.11. TÉTEL: A SZEKVENCIA LEVEZETÉSI SZABÁLYÁNAK MEGFORDÍTÁSA
Legyen $S = (S_1; S_2)$, és Q, R olyan állítások A-n, amelyekre $Q \Rightarrow lf(S, R)$. Ekkor $\exists Q' : A \to \mathbb{L}$ állítás, amire

(1) $Q \Rightarrow lf(S_1, Q')$ és

(2) $Q' \Rightarrow lf(S_2, R)$.

Bizonyítás: Legyen $Q' = lf(S_2, R)$. Ekkor (2) automatikusan teljesül. Csak (1) fennállását kell belátnunk. Ehhez indirekt feltesszük, hogy

$$\exists q \in \lceil Q \rceil : q \notin \lceil lf(S_1, lf(S_2, R)) \rceil.$$

Ez kétféleképpen fordulhat elő:

- $q \notin \mathcal{D}_{p(S_1)}$, de ez ellentmond annak a feltételnek, mely szerint $\lceil Q \rceil \subseteq \mathcal{D}_{p(S)} \subseteq \mathcal{D}_{p(S_1)}$.
- $p(S_1)(q) \not\subseteq \lceil lf(S_2, R) \rceil$. Ekkor legyen $r \in p(S_1)(q) \setminus \lceil lf(S_2, R) \rceil$. Ekkor két eset lehetséges:
  - $r \notin \mathcal{D}_{p(S_2)}$. Ez ellentmond annak, hogy $q \in \mathcal{D}_{p(S_2) \odot p(S_1)}$.
  - $p(S_2)(r) \not\subseteq \lceil R \rceil$: Legyen $s \in p(S_2)(r) \setminus \lceil R \rceil$. Ekkor $s \in p(S)(q)$ és $s \notin \lceil R \rceil$, és ez ellentmond a $p(S)(q) \subseteq \lceil R \rceil$ feltételnek.

□

Vizsgáljuk meg, mi a kapcsolat az elágazás és az őt alkotó programok adott utófeltételhez tartozó leggyengébb előfeltétele között.

7.12. TÉTEL: AZ ELÁGAZÁS LEVEZETÉSI SZABÁLYA
Legyen $IF = (\pi_1 : S_1, \ldots, \pi_n : S_n)$, és adott Q, R állítás A-n. Ha

$$\forall i \in [1..n] : \; Q \wedge \pi_i \Rightarrow lf(S_i, R),$$

akkor

$$Q \wedge \left(\bigvee_{i=1}^{n} \pi_i \right) \Rightarrow lf(IF, R).$$

Bizonyítás: Legyen $q \in \lceil Q \rceil$, és tegyük fel, hogy valamelyik feltétel igaz q-ra, azaz $\exists i \in [1..n] : \pi_i(q)$. Ekkor $q \in \mathcal{D}_{p(IF)}$, ui.

$$\forall j \in [1..n] : q \in \pi_j \Rightarrow q \in \lceil lf(S_j, R) \rceil \Rightarrow q \in \mathcal{D}_{p(S_j)}.$$

Másrészt mivel $\forall j \in [1..n] : \pi_j(q) \Rightarrow p(S_j)(q) \subseteq \lceil R \rceil$:

$$p(IF)(q) = \bigcup_{\substack{j \in [1..n] \\ \pi_j(q)}} p(S_j)(q) \subseteq \lceil R \rceil,$$

azaz $q \in \lceil lf(IF, R) \rceil$.

□

Felhasználva a specifikáció tételét és az elágazás levezetési szabályát azt mondhatjuk: Legyen adott az F feladat specifikációja (A, B, Q, R). Ekkor ha minden $b \in B$ paraméterre és minden S_i programra $Q_b \wedge \pi_i \Rightarrow lf(S_i, R_b)$, és minden b paraméterhez

van olyan π_i feltétel, amelyre $Q_b \Rightarrow \pi_i$, akkor az elágazás megoldja a Q_b, R_b párokkal specifikált feladatot.

Hasonlóan a szekvencia levezetési szabályához az elágazás levezetési szabálya is megfordítható, tehát ha egy elágazás megold egy specifikált feladatot, akkor le is lehet vezetni.

7.13. TÉTEL: AZ ELÁGAZÁS LEVEZETÉSI SZABÁLYÁNAK MEGFORDÍTÁSA

Legyen $IF = (\pi_1 : S_1, \ldots, \pi_n : S_n)$, és Q, R olyan állítások A-n, amelyekre

$$Q \wedge \left(\bigvee_{i=1}^{n} \pi_i \right) \Rightarrow lf(IF, R).$$

Ekkor

$$\forall i \in [1..n] : \; Q \wedge \pi_i \Rightarrow lf(S_i, R).$$

Bizonyítás: Indirekt: tegyük fel, hogy

$$\exists i \in [1..n] : \lceil Q \wedge \pi_i \rceil \not\subseteq \lceil lf(S_i, R) \rceil.$$

Legyen $q \in \lceil Q \wedge \pi_i \rceil \setminus \lceil lf(S_i, R) \rceil$. Ekkor két eset lehetséges:

- $q \notin \mathcal{D}_{p(S_i)}$. Ez ellentmond annak a feltevésnek, hogy $q \in \mathcal{D}_{p(IF)}$.
- $p(S_i)(q) \not\subseteq \lceil R \rceil$. Ekkor

  $$p(S_i)(q) \subseteq p(IF)(q) \subseteq \lceil R \rceil$$

  tehát ellentmondásra jutottunk.

□

Természetesen nem elég az elágazás levezetési szabályának teljesülését vizsgálni, ha arra vagyunk kíváncsiak, hogy az elágazás megold-e egy feladatot. Soha nem szabad elfeledkeznünk arról, hogy leellenőrizzük, hogy a feltételek lefedik-e a feladat értelmezési tartományát.

Ha $\forall b \in B : Q_b \Rightarrow \bigvee_{i=1}^{n} \pi_i$, akkor a levezetési szabály teljesülése – a specifikáció tétele alapján – garantálja, hogy az elágazás megoldja a feladatot.

Az előző két konstrukcióhoz hasonlóan most megvizsgáljuk, hogy milyen kapcsolat van a ciklus és a ciklusmag leggyengébb előfeltétele között.

7.14. TÉTEL: A CIKLUS LEVEZETÉSI SZABÁLYA

Adott P, Q, R állítás A-n, $t : A \to \mathbb{Z}$ 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) $P \wedge \pi \Rightarrow lf(S_0, P)$

(5) $P \wedge \pi \wedge t = t_0 \Rightarrow lf(S_0, t < t_0)$

akkor

$$Q \Rightarrow lf(DO, R).$$

Jelölje a továbbiakban g a ciklusmag programfüggvényének a ciklusfeltételre vonatkozó leszűkítését, azaz $g = p(S_0)|_{\lceil\pi\rceil}$. Mielőtt magát a levezetési szabályt bizonyítanánk, ennek a g relációnak látjuk be két tulajdonságát:
1. állítás: Legyen $q \in \lceil P \wedge \pi \rceil$. Ekkor

$$\forall k \in \mathbb{N}_0 : g^k(q) \subseteq \lceil P \rceil.$$

Bizonyítás: k szerinti teljes indukcióval:

- $k = 0$: $g^0(q) = q \in \lceil P \rceil$
- Tegyük fel, hogy $g^k(q) \subseteq \lceil P \rceil$. Legyen $r \in g^k(q)$ tetszőleges. Ekkor két eset lehetséges:
  - $r \in \lceil P \wedge \neg\pi \rceil$. Ekkor $g(r) = r \in \lceil P \rceil$.
  - $r \in \lceil P \wedge \pi \rceil$. Ekkor (4) miatt $r \in \mathcal{D}_{p(S_0)} \wedge g(r) = p(S_0)(r) \subseteq \lceil P \rceil$.

  Tehát: $g^{k+1}(q) \subseteq \lceil P \rceil$.

□
2. állítás: Legyen $q \in \lceil P \wedge \pi \rceil$. Ekkor

$$\exists n \leq t(q) : g^n(q) \subseteq \lceil \neg\pi \rceil.$$

Bizonyítás: Indirekt: tegyük fel, hogy $\forall n \in \mathbb{N} : g^n(q) \subseteq \lceil \pi \rceil$. Tekintsünk egy tetszőleges $b \in p(S_0)(q)$ pontot. Ekkor $t(b) < t(q)$, ugyanis (5)-ben t_0 helyére $t(q)$-t írva: $t(b) < t(q)$ teljesül. Ekkor viszont

$$\max_{b \in g^{t(q)+1}(q)} t(b) < \max_{b \in g^{t(q)}(q)} t(b) < \cdots < \max_{b \in g(q)} t(b) < t(q),$$

azaz

$$\max_{b \in g^{t(q)+1}(q)} t(b) < 0.$$

Ez viszont ellentmond (3)-nak.
□
Bizonyítás: *(Ciklus levezetési szabálya)* Legyen $q \in \lceil Q \rceil$ tetszőleges. Mivel (1) miatt $\lceil Q \rceil \subseteq \lceil P \rceil$, ezért $q \in \lceil P \wedge \neg\pi \rceil$ vagy $q \in \lceil P \wedge \pi \rceil$ teljesül.

- Tegyük fel, hogy $q \in \lceil P \wedge \neg\pi \rceil$. Ekkor a ciklus definíciója alapján $p(DO)(q) = \{q\}$, másrészt (2) miatt $q \in \lceil R \rceil$, tehát $q \in \lceil lf(DO, R) \rceil$.
- Tekintsük most azt az esetet, amikor $q \in \lceil P \wedge \pi \rceil$ teljesül. Ekkor a 2. állítás alapján

  $$\exists n \in \mathbb{N}_0 : (p(S_0)|_\pi)^n(q) = \emptyset,$$

  azaz

  $$q \in \mathcal{D}_{p(DO)}.$$

  Ekkor a feltételre vonatkozó lezárt definíciója alapján:

  $$p(DO)(q) \subseteq \lceil \neg\pi \rceil.$$

  Másrészt az 1. állítás miatt

  $$p(DO)(q) \subseteq \lceil P \rceil,$$

és ekkor (2) miatt

$$p(DO)(q) \subseteq \lceil P \wedge \neg\pi \rceil \subseteq \lceil R \rceil,$$

tehát

$$q \in \lceil lf(DO, R) \rceil.$$

□

A levezetési szabályban szereplő P állítást a ciklus invariáns tulajdonságának, a t függvényt terminálófüggvénynek nevezzük. A P invarianciáját az (1) és (4) feltételek biztosítják: garantálják, hogy az invariáns tulajdonság a ciklusmag minden lefutása előtt és után teljesül. A terminálófüggvény a ciklus befejeződését biztosítja: az (5) pont alapján a ciklusmag minden lefutása legalább eggyel csökkenti a terminálófüggvényt, másrészt a (3) miatt a terminálófüggvénynek pozitívnak kell lennie. A (2) feltétel garantálja, hogy ha a ciklus befejeződik, akkor az utófeltétel igazsághalmazába jut.

A ciklus levezetési szabályának és a specifikáció tételének felhasználásával elégséges feltételt adhatunk a megoldásra: ha adott a feladat specifikációja (A, B, Q, R), és találunk olyan invariáns állítást és terminálófüggvényt, hogy a paramétertér minden elemére teljesül a ciklus levezetési szabályának öt feltétele, akkor a ciklus megoldja a (Q_b, R_b) párokkal definiált feladatot.

A ciklus levezetési szabálya visszafelé nem igaz, azaz van olyan ciklus, amit nem lehet levezetni. Ennek az az oka, hogy egy levezetett ciklus programfüggvénye mindig korlátos lezárt, hiszen az állapottér minden pontjában a terminálófüggvény értéke korlátozza a ciklusmag lefutásainak számát.

Az is könnyen látható, hogy ha a ciklus programfüggvénye nem korlátos lezárt, akkor nem tudunk a ciklushoz terminálófüggvényt adni. Ám ha egy ciklus programfüggvénye megegyezik a ciklusmag ciklusfeltételre vonatkozó korlátos lezártjával, akkor a ciklus levezethető.

7.15. TÉTEL: A CIKLUS LEVEZETÉSI SZABÁLYÁNAK MEGFORDÍTÁSA

Legyen $DO = (\pi, S_0)$, és Q, R olyan állítások A-n, amelyekre $Q \Rightarrow lf(DO, R)$, és tegyük fel, hogy $p(DO) = \overline{\overline{p(S_0)|_\pi}}$. Ekkor létezik P állítás és $t : A \to \mathbb{Z}$ függvény, amelyekre

(1) $Q \Rightarrow P$

(2) $P \wedge \neg\pi \Rightarrow R$

(3) $P \wedge \pi \Rightarrow t > 0$

(4) $P \wedge \pi \Rightarrow lf(S_0, P)$

(5) $P \wedge \pi \wedge t = t_0 \Rightarrow lf(S_0, t < t_0)$

Bizonyítás: Legyen $P = lf(DO, R)$ és válasszuk t-t úgy, hogy $\forall a \in A$:

$$t(a) = \begin{cases} 0, & \text{ha } a \notin \mathcal{D}_{p(DO)} \cap \lceil \pi \rceil \\ max\{i \in \mathbb{N} \mid p(S_0)^i(a) \cap \lceil \pi \rceil \neq \emptyset\}, & \text{ha } a \in \mathcal{D}_{p(DO)} \cap \lceil \pi \rceil \end{cases}$$

Ekkor

(1) $Q \Rightarrow lf(DO, R)$ triviálisan teljesül.

(2) Legyen $a \in \lceil P \wedge \neg\pi \rceil$. Ekkor mivel $a \in \lceil lf(DO, R) \rceil$, $p(DO)(a) \subseteq \lceil R \rceil$. Másrészt mivel $a \in \lceil \neg\pi \rceil$, $p(DO)(a) = a$. Tehát $a \in \lceil R \rceil$, azaz $\lceil P \wedge \neg\pi \rceil \subseteq \lceil R \rceil$.

(3) t definíciója miatt nyilvánvaló.

(4) Felhasználva a lezárt azon egyszerű tulajdonságát, hogy

$$a \in \mathcal{D}_{\overline{R}} \Longrightarrow R(a) \subseteq \mathcal{D}_{\overline{R}}$$

a következő eredményt kapjuk: Legyen $a \in \lceil lf(DO, R) \wedge \pi \rceil$. Ekkor

$$a \in \mathcal{D}_{p(S_0)}, \quad \text{és} \quad p(S_0)(a) \subseteq \lceil lf(DO, R) \rceil,$$

tehát

$$a \in lf(S_0, P).$$

(5) A t definíciójából leolvasható, hogy a ciklusmag egyszeri végrehajtása csökkenti a terminálófüggvény értékét:

Legyen $a \in \lceil P \wedge \pi \rceil$, $t_0 = t(a)$, $b \in p(S_0)(a)$. Ekkor ha

$$t(a) = max\{i \in \mathbb{N} \mid p(S_0)^i(a) \cap \lceil \pi \rceil \neq \emptyset\}$$

akkor

$$t(b) < t(a)$$

azaz

$$t(b) < t_0.$$

□

Azt, hogy a lezárt és a korlátos lezárt megegyezik, a t definíciójában használtuk ki: ez a feltétel garantálja, hogy a definícióban szereplő maximum véges.

7.4. Példák

7.5. Feladatok

7.1. $A = \{1, 2, 3, 4, 5, 6\}$. $\lceil \pi_1 \rceil = \{1, 2, 3, 4\}$. $\lceil \pi_2 \rceil = \{1, 3, 4, 5\}$.

$$\begin{array}{lllll} S_1 = \{ & 1\to 14 & 1\to 12\ldots & 2\to 2132 & 3\to 36 \\ & 4\to 463 & 4\to 451 & 5\to 563 & 6\to 612 \quad \} \\ S_2 = \{ & 1\to 134 & 1\to 121 & 2\to 2132\ldots & 3\to 36 \\ & 4\to 463 & 4\to 451\ldots & 5\to 5632 & 6\to 61\ldots \quad \} \end{array}$$

Add meg az $(S_1; S_2)$, $IF(\pi_1 : S_1, \pi_2 : S_2)$, $DO(\pi_1, S_1)$ programokat és a programfüggvényeiket!

7.2. $A = \{1, 2, 3, 4, 5, 6\}$, $\lceil \pi_1 \rceil = \{1, 2, 3, 4\}$, $\lceil \pi_2 \rceil = \{2, 3, 4\}$, $\lceil \pi_3 \rceil = \{1, 4, 6\}$.

$$\begin{array}{lllll} S_1 = \{ & 1\to 12\ldots & 2\to 23 & 3\to 3456 & \\ & 4\to 463 & 5\to 53 & 6\to 62 & \} \\ S_2 = \{ & 1\to 12 & 2\to 24 & 3\to 3\ldots & \\ & 4\to 43 & 5\to 5 & 6\to 61 & \} \\ S_3 = \{ & 1\to 12 & 2\to 2\ldots & 3\to 31 & \\ & 4\to 432 & 5\to 5\ldots & 6\to 63\ldots & \} \end{array}$$

$IF(\pi_1 : S_1, \pi_2 : S_2, \pi_3 : S_3) =? \; \mathcal{D}_{p(IF)} =? \; p(IF) =?$

7.3. Fejezzük ki a SKIP ill. az ABORT programot egy tetszőleges S program és a programkonstrukciók segítségével!

7.4. Legyen S_1 és S_2 egy-egy program az A állapottéren. Igaz-e, hogy $S_2 \circ \tau \circ S_1$ megegyezik $(S_1; S_2)$-vel?

7.5. $S = (S_1; S_2)$. Igaz-e, hogy

a) $\mathcal{D}_{p(S)} = \lceil lf(S_1, \mathcal{P}(\mathcal{D}_{p(S_2)}))\rceil$

b) tetszőleges R utófeltételre: $lf((S_1; S_2), R) = lf(S_1, lf(S_2, R))$?

7.6. Van-e olyan program, ami felírható szekvenciaként is, elágazásként is, és felírható ciklusként is?

7.7. Igaz-e, hogy minden program felírható szekvenciaként is, elágazásként is, és felírható ciklusként is?

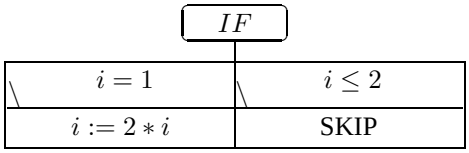
7.8. $IF = (\pi_1 : S_1, \ldots, \pi_n : S_n)$. Igaz-e, hogy $\mathcal{D}_{p(IF)} = \bigcup_{k=1}^{n} (ection\pi_k \cap \mathcal{D}_{p(S_k)})$?

7.9. Legyen $S_1, S_2, \ldots, S_n$ program A-n! $IF = (\pi_1 : S_1, \ldots, \pi_n : S_n)$. $S = S_1 \cup S_2 \cup \cdots \cup S_n$. Keressünk olyan π_k feltételeket és S_k programokat, hogy $\mathcal{D}_{p(IF)} = A$ és $\mathcal{D}_{p(S)} = \emptyset$!

7.10. $IF = (\pi_1 : S_1, \ldots, \pi_n : S_n)$. $S = S_1 \cup S_2 \cup \cdots \cup S_n$. Igaz-e, hogy $p(IF)$ része $p(S)$-nek?

7.11. Igaz-e? Ha $IF = (\pi_1 : S_1, \pi_2 : S_2)$, akkor $\mathcal{D}_{p(IF)} = (\lceil\pi_1\rceil \cap \lceil\pi_2\rceil \cap \mathcal{D}_{p(S_1)} \cap \mathcal{D}_{p(S_2)}) \cup (\mathcal{D}_{p(S_1)} \cap (\lceil\pi_1\rceil \setminus \lceil\pi_2\rceil)) \cup (\mathcal{D}_{p(S_2)} \cap (\lceil\pi_2\rceil \setminus \lceil\pi_1\rceil))$?

7.12. $A = \{1, 2, 3, 4\}$.



Milyen sorozatokat rendel S_1, S_2, IF az állapottér egyes pontjaihoz?

7.13. $S = (S_1; S_2)$. S_1 megoldja F_1-et és S_2 megoldja F_2-t. Megoldja-e S az

a) $F = F_2 \circ F_1$

b) $F = F_2 \odot F_1$ feladatot?

7.14. $S = (S_1; S_2)$. S megoldása az $(F_2 \odot F_1)$ feladatnak.
Megoldja-e S_1 az F_1-et ill. S_2 az F_2-t?

7.15. $IF = (\pi_1 : S_1, \ldots, \pi_n : S_n)$. $F \subseteq A \times A$ feladat. $\forall k \in [1, n] :$ S_k megoldja az $F|_{\lceil\pi_k\rceil}$ feladatot.

a) IF megoldja-e az F feladatot?

b) IF megoldja-e az F feladatot, ha $\pi_1 \vee \pi_2 \vee \cdots \vee \pi_n =$ igaz?

c) IF megoldja-e az F feladatot, ha $\mathcal{D}_F \subseteq \lceil\pi_1 \vee \pi_2 \vee \cdots \vee \pi_n\rceil$?

7.16. $IF = (\pi_1 : S_1, \ldots, \pi_n : S_n)$. $F \subseteq A \times A$ feladat. IF megoldja az F feladatot. Igaz-e, hogy $\forall k \in [1,n] :\ S_k$ megoldja az $F|_{\lceil \pi_k \rceil}$ feladatot?

7.17. $IF = (\pi_1 : S_1, \ldots, \pi_n : S_n)$. $F_1, \ldots, F_k \subseteq A \times A$ feladat. $\forall k \in [1,n] :\ S_k$ megoldja az F_k feladatot. Megoldja-e IF az $F = F_1 \cup \cdots \cup F_n$ feladatot?

7.18. $IF = (\pi_1 : S_1, \ldots, \pi_n : S_n)$. $F \subseteq A \times A$ feladat. IF megoldja az F feladatot és $\lceil \pi_1 \rceil \cup \cdots \cup \lceil \pi_n \rceil \subseteq \mathcal{D}_F$. Igaz-e, hogy $\forall k \in [1,n] :\ S_k$ megoldja az $F|_{\lceil \pi_k \rceil}$ feladatot.

7.19. $IF = (\pi_1 : S_1, \ldots, \pi_n : S_n)$. $F_1, \ldots, F_n \subseteq A \times A$ feladat. $\forall k \in [1,n] :\ \mathcal{D}_{F_k} \subseteq \lceil \pi_k \rceil$ és S_k megoldja az F_k feladatot. $F = F_1 \cup \cdots \cup F_n$. Megoldja-e IF az F feladatot?

7.20. Igaz-e, hogy $IF_1 = (\pi_1 : S_1, \pi_2 : S_2)$ és $IF_2 = (\pi_1 : S_1, \pi_1 \wedge \pi_2 : S_1 \cup S_2, \pi_2 : S_2)$

a) egyenlő?

b) ekvivalens?

7.21. Legyen $IF_{34} = (\pi_3 : S_3, \pi_4 : S_4)$, $IF_1 = (\pi_1 : S_1, \pi_2 : IF_{34})$, $IF_2 = (\pi_1 : S_1, \pi_2 \wedge \pi_3 : S_3, \pi_2 \wedge \pi_4 : S_4)$! Igaz-e, hogy IF_1 és IF_2

a) egyenlő?

b) ekvivalens?

7.22. $F \subseteq A \times A$ feladat. S_0 program A-n. S_0 megoldja F-et. Megoldja-e a $DO(\pi, S_0)$ program az F feladat π-re vonatkozó lezártját?

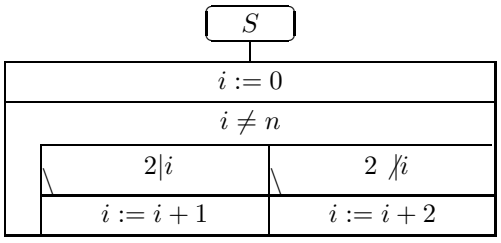
7.23. Legyen $DO = (\pi, S)$! Igaz-e, hogy

a) $p(DO) \subseteq p(S)$?

b) $p(S) \subseteq p(DO)$?

7.24. $A = \underset{i}{\mathbb{N}_0} \times \underset{n}{\mathbb{N}_0}$

$S = ((i := 0; DO(i \neq n, IF(\ 2|i :\ i := i+1,\ \ 2 \not| i\ :\ \ i := i+2))))$



Milyen sorozatokat rendel S a $(2,4)$ ill. a $(3,7)$ ponthoz?

7.25. Tegyük fel, hogy teljesül a ciklus levezetési szabályának mind az öt feltétele, és Q igazsághalmaza nem üres. Lehet-e üres a

a) $\lceil P \wedge R \rceil$

b) $\lceil P \wedge \neg\pi \wedge R \rceil$ halmaz?

7.26. Tegyük fel, hogy teljesül a ciklus levezetési szabályának mind az öt feltétele, és $(Q \wedge \pi)$ igazsághalmaza nem üres. Lehet-e üres a $lf(S_0, P)$ és $lf(DO, R)$ igazsághalmazának metszete?

7.27. Tegyük fel, hogy teljesül a ciklus levezetési szabályának mind az öt feltétele. Legyen $g = p(S_0) \cap (\lceil \pi \rceil \times A)$ és $q \in \lceil P \rceil \cap \lceil \pi \rceil$. Igaz-e, hogy

a) $\forall k \in \mathbb{N}_0 : \; g^k(q) \subseteq \lceil P \rceil$

b) $b \in g^k(q) \cap \lceil \pi \rceil \cap \lceil P \rceil \Rightarrow t(b) \leq t(q) - k$?

c) $g|_\pi = p(S_0)|_\pi$?

d) $\exists k \in \mathbb{N}_0 : \; k \leq t(q) \wedge g^k(q) \subseteq \lceil \neg \pi \rceil$?

7.28. Legyen $S = (S_1; S_2)$ és Q, Q' és R olyan állítások, amelyekre
$Q \Rightarrow lf(S, R), Q' \Rightarrow lf(S_2, R), Q \Rightarrow lf(S_1, Q')$.
Lehetséges-e, hogy $\lceil Q \rceil \cap \lceil R \rceil = \emptyset \wedge \lceil Q \rceil \cap \lceil Q' \rceil \neq \emptyset \wedge \lceil Q' \rceil \cap \lceil R \rceil \neq \emptyset$?
Indokold, ha nem, és írj rá példát, ha igen!

7.29.
$$\begin{array}{rl} A = & \underset{z}{\mathbb{Z}} \times \underset{y}{\mathbb{N}_0} \qquad B = \underset{z'}{\mathbb{Z}} \times \underset{y'}{\mathbb{N}_0} \\ & Q = (x = x' \wedge y = y') \\ & R = (x = x' - y' \wedge y = 0) \\ S_0 = & \{((x,y), <(x,y), (x-1,y), (x-1,y-1)>) \mid x \in \mathbb{Z} \text{ és } y \in \mathbb{N}\} \cup \\ & \{((x,0), <(x,0)>) \mid x \in \mathbb{Z}\} \\ DO = & \{ ((x,y), <(x,y), (x-1,y), (x-1,y-1), (x-2,y-1), \\ & (x-2,y-2), \ldots, (x-y+1,1), (x-y,1)(x-y,0)>) \mid x \in \mathbb{Z} \text{ és } y \in N_0\} \end{array}$$
Megjegyzés: Az $(x, 0)$ párhoz 1 hosszúságú, az $(x, 1)$ párhoz 3 hosszúságú, az $(x, 2)$ párhoz 5 hosszúságú sorozatot rendel a program.
Tudjuk, hogy $DO = (\pi; S_0)$ valamilyen π-re. Igaz-e, hogy található olyan P állítás és $t : A \to \mathbb{Z}$ függvény, hogy a ciklus levezetési szabályának feltételei teljesülnek, és ha igen, adj meg egy megfelelő π-t, P-t és t-t!

7.30.
$$\begin{array}{rl} A = & \underset{k}{\mathbb{Z}} \times \underset{x}{\mathbb{Z}} \times \underset{i}{\mathbb{Z}} \times \underset{a}{\mathbb{Z}} \times \underset{b}{\mathbb{Z}} \qquad B = \underset{a'}{\mathbb{Z}} \times \underset{b'}{\mathbb{Z}} \end{array}$$
$S = (k := 5; ((a > b : x := a - b, a \leq b : x := b - a); i := i + 1))$
$Q = (a = a' \wedge b = b' \wedge i \in [0, 1] \wedge |a - b| > 10)$
$R = (a = a' \wedge b = b' \wedge k * i \leq x)$
Bizonyítsuk be, hogy $Q \Rightarrow lf(S, R)$!

7.6. A programkonstrukciók és a kiszámíthatóság

Ebben az alfejezetben kis kitérőt teszünk a kiszámíthatóság-elmélet felé, és megmutatjuk, hogy az imént bevezetett három programkonstrukció segítségével minden – elméletileg megoldható – feladatot meg tudunk oldani. Ehhez kapcsolatot létesítünk a kiszámítható függvények és a „jól konstruált” programok között.

7.6.1. Parciális rekurzív függvények

A Church tézis szerint a kiszámítható függvények halmaza megegyezik a parciális rekurzív függvények halmazával. A kiszámíthatóság ezen modelljében csak $f \in \mathbb{N}^m \to \mathbb{N}^n$ típusú függvények szerepelnek. Először az alapfüggvényeket definiáljuk:

- $suc : \mathbb{N} \to \mathbb{N}, \quad \forall x \in \mathbb{N}$:
$$suc(x) = x + 1,$$
- $\forall n \in \mathbb{N}_0 : \quad c_1^{(n)} : \mathbb{N}^n \to \mathbb{N}, \quad \forall (x_1, \ldots, x_n) \in \mathbb{N}^n$:
$$c_1^{(n)}(x_1, \ldots, x_n) = 1,$$
- $\forall n \in \mathbb{N} : \forall i \in [1..n] : \quad pr_i^{(n)} : \mathbb{N}^n \to \mathbb{N}, \quad \forall (x_1, \ldots, x_n) \in \mathbb{N}^n$:
$$pr_i^{(n)}(x_1, \ldots, x_n) = x_i.$$

A továbbiakban definiálunk néhány elemi függvény-operátort:

Kompozíció Legyen $f \in \mathbb{N}^m \to \mathbb{N}^n$ és $g \in \mathbb{N}^n \to \mathbb{N}^k$. Az f és g kompozíciója az alábbi függvény:
$g \circ f \in \mathbb{N}^m \to \mathbb{N}^k, \quad \mathcal{D}_{g \circ f} = \{x \in \mathcal{D}_f \mid f(x) \in \mathcal{D}_g\}$ és $\forall x \in \mathcal{D}_{g \circ f}$:
$$(g \circ f)(x) = g(f(x)).$$

Vegyük észre, hogy ez az operátor megegyezik az alapfogalmaknál bevezetett relációk közötti kompozícióval (tulajdonképpen a szigorú kompozícióval, de függvények esetén ez a kettő azonos).

Unió Legyen $k \in \mathbb{N}$ rögzített, és $\forall i \in [1..k] : f_i \in \mathbb{N}^m \to \mathbb{N}^{n_i}$. E függvények uniója $(f_1, f_2, \ldots, f_k) \in \mathbb{N} \to \mathbb{N}^{n_1} \times \cdots \times \mathbb{N}^{n_k}, \mathcal{D}_{(f_1, f_2, \ldots, f_k)} = \bigcap_{i=1}^{k} \mathcal{D}_{f_i}$ és $\forall x \in \mathcal{D}_{(f_1, f_2, \ldots, f_k)}$:
$$(f_1, f_2, \ldots, f_k)(x) = (f_1(x), \ldots, f_k(x)).$$

Rekurzió Legyen n rögzített, $f \in \mathbb{N}^n \to \mathbb{N}$ és $g \in \mathbb{N}^{n+2} \to \mathbb{N}$. Az f függvény g szerinti rekurziója $\varrho(f, g) \in \mathbb{N}^{n+1} \to \mathbb{N}$, és
$$\begin{aligned} \varrho(f,g)(x_1, \ldots, x_n, 1) &= f(x_1, \ldots, x_n), \\ \varrho(f,g)(x_1, \ldots, x_n, k+1) &= g(x_1, \ldots, x_n, k, \varrho(f,g)(x_1, \ldots, x_n, k)). \end{aligned}$$

A függvényhez hasonlóan rekurzívan definiálhatnánk ennek a függvénynek az értelmezési tartományát, de ez kívül esik a jelenlegi vizsgálódásunk körén. Ezért csak azt mondjuk, hogy az értelmezési tartomány azon pontok halmaza, ahonnét kiindulva a fenti rekurzió elvégezhető.

μ-operátor Legyen $f \in \mathbb{N}^{n+1} \to \mathbb{N}$. A μ-operátort erre a függvényre alkalmazva azt a $\mu(f) \in \mathbb{N}^n \to \mathbb{N}$ függvényt kapjuk, amelyre $\mathcal{D}_{\mu(f)} = \{(x_1, \ldots, x_n) \in \mathbb{N}^n \mid \exists y \in \mathbb{N} : \ f(x_1, \ldots, x_n, y) = 1 \wedge \forall i \in [1..y-1] : \ (x_1, \ldots, x_n, i) \in \mathcal{D}_f\}$, és $\forall (x_1, \ldots, x_n) \in \mathcal{D}_{\mu(f)}$:

$$mu(f)(x_1, \ldots, x_n) = min\{y \mid f(x_1, \ldots, x_n, y) = 1\}.$$

A fenti alapfüggvények és a bevezetett operátorok segítségvel már definiálható a parciális rekurzív függvények halmaza.

7.4. Definíció (PARCIÁLIS REKURZÍV FÜGGVÉNY). *Az $f : \mathbb{N}^m \to \mathbb{N}^n (n, m \in \mathbb{N})$ függvény akkor és csak akkor parciális rekurzív, ha az alábbiak egyike teljesül:*

- *f az alapfüggvények valamelyike*
- *f kifejezhető a fenti operátorok parciális rekurzív függvényekre történő alkalmazásával.*

7.6.2. A parciális rekurzív függvények kiszámítása

Ahhoz, hogy a fenti függvényeket kiszámító programokat tudjunk adni, definiálnunk kell az ilyen függvények által meghatározott feladatot. Legyen $f : \mathbb{N}^m \to \mathbb{N}^n$ egy tetszőleges függvény (m, n rögzített). Az f által meghatározott feladat specifikációja:

$$\begin{array}{lccccccc} A = & \mathbb{N} & \times \ldots \times & \mathbb{N} & \times & \mathbb{N} & \times \ldots \times & \mathbb{N} \\ & x_1 & & x_m & & y_1 & & y_m \\ B = & \mathbb{N} & \times \ldots \times & \mathbb{N} & & & & \\ & x'_1 & & x'_m & & & & \end{array}$$

$$Q : (x_1 = x'_1 \wedge \cdots \wedge x_m = x'_m \wedge (x_1, \ldots, x_m) \in \mathcal{D}_f)$$

$$R : (Q \wedge (y_1, \ldots, y_n) = f(x'_1, \ldots, x'_m))$$

Most megmutatjuk, hogy minden parciális rekurzív függvény által meghatározott feladat megoldható „jól konstruált” programmal (jól konstruáltnak tekintünk egy programot, ha elemi programokból a fenti három konstrukcióval megkapható). A bizonyításhoz csak annyit kell feltételeznünk, hogy az alapfüggvények kiszámíthatók (megengedett) elemi programokkal.

A fenti feltételezést felhasználva elegendő azt megmutatni, hogy az elemi függvényoperátorok (kompozíció, unió, rekurzió, μ-operátor) kiszámíthatók jól konstruált programokkal.

Kompozíció. Legyen $f \in \mathbb{N}^m \to \mathbb{N}^n$ és $g \in \mathbb{N}^n \to \mathbb{N}^k$. Ekkor az $g \circ f$ által meghatározott feladat specifikációja:

$$\begin{array}{lccccccc} A = & \mathbb{N} & \times \ldots \times & \mathbb{N} & \times & \mathbb{N} & \times \ldots \times & \mathbb{N} \\ & x_1 & & x_m & & y_1 & & y_k \\ B = & \mathbb{N} & \times \ldots \times & \mathbb{N} & & & & \\ & x'_1 & & x'_m & & & & \end{array}$$

$$Q : (x_1 = x'_1 \wedge \cdots \wedge x_m = x'_m \wedge (x_1, \ldots, x_m) \in \mathcal{D}_{g \circ f})$$

$$R : (Q \wedge (y_1, \ldots, y_n) = (g \circ f)(x'_1, \ldots, x'_m))$$

Jelöljük $z_1, \ldots, z_n := f(x_1, \ldots, x_m)$-mel azt a programot, amely kiszámítja f-et, és hasonlóan $y_1, \ldots, y_k := g(z_1, \ldots, z_n)$-nel azt, amelyik kiszámítja g-t. Tegyük fel, hogy ez a két program vagy elemi értékadás, vagy jól konstruált program. Ebben az

esetben a két program szekvenciája kiszámítja $g \circ f$-et, azaz megoldja a fent specifikált feladatot. Legyen Q' a második program R utófeltételhez tartozó leggyengébb előfeltétele:

$$Q' : (Q \wedge g(z_1, \ldots, z_n) = (g \circ f)(x'_1, \ldots, x'_m) \wedge (z_1, \ldots, z_n) \in \mathcal{D}_g)$$

Most vizsgáljuk meg az első program ezen Q' utófeltételhez tartozó leggyengébb előfeltételét:

$$lf(z_1, \ldots, z_n := f(x_1, \ldots, x_m), Q') = (Q \wedge g(f(x_1, \ldots, x_m))) = \\ ((g \circ f)(x'_1, \ldots, x'_m) \wedge f(x_1, \ldots, x_m) \in \mathcal{D}_g \wedge (x_1, \ldots, x_m) \in \mathcal{D}_f)$$

Könnyen látható, hogy ez a leggyengébb előfeltétel következik Q-ból, és így a szekvencia levezetési szabálya és a specifikáció tételének alkalmazásával beláttuk, hogy a

$z_1, \ldots, z_n := f(x_1, \ldots, x_m)$
$y_1, \ldots, y_k := g(z_1, \ldots, z_n)$

program megoldja a fent specifikált feladatot, azaz kiszámítja f és g kompozícióját.

Unió. Legyen $k \in \mathbb{N}$ rögzített, és $\forall i \in [1..k] : f_i \in \mathbb{N}^m \to \mathbb{N}^{n_i}$. Ekkor az e függvények uniója által meghatározott feladat specifikációja:

$$A = \underset{x_1}{\mathbb{N}} \times \ldots \times \underset{x_m}{\mathbb{N}} \times \\ \times \underset{y_{1\,1}}{\mathbb{N}} \times \ldots \times \underset{y_{1\,n_1}}{\mathbb{N}} \\ \vdots \\ \times \underset{y_{k\,1}}{\mathbb{N}} \times \ldots \times \underset{y_{k\,n_k}}{\mathbb{N}}$$

$$B = \underset{x'_1}{\mathbb{N}} \times \ldots \times \underset{x'_m}{\mathbb{N}}$$

$$Q : (x_1 = x'_1 \wedge \cdots \wedge x_m = x'_m \wedge (x_1, \ldots, x_m) \in \mathcal{D}_{(f_1, \ldots, f_k)})$$

$$R : (Q \wedge (y_{1\,1}, \ldots, y_{1\,n_1} \ldots y_{k1}, \ldots, y_{k\,n_k}) = (f_1, \ldots, f_k)(x'_1, \ldots, x'_m))$$

Tegyük fel, hogy a komponens függvények kiszámíthatók jól konstruált programmal, vagy elemi értékadással. Jelölje $y_{i1}, \ldots, y_{i\,n_i} := f_i(x_1, \ldots, x_m)$ az i-edik függvényt $(1 \leq i \leq k)$ kiszámító programot. Ekkor ezeknek a programoknak a szekvenciája megoldja a fenti feladatot. Legyen Q_k a k-adik program R utófeltételhez tartozó leggyengébb előfeltételét:

$$Q_k : (Q \wedge (y_{1\,1}, \ldots, y_{1\,n_1} \ldots y_{k-1\,1}, \ldots, y_{k-1\,n_{k-1}}, f_k(x_1, \ldots, x_m)) = \\ (f_1, \ldots, f_k)(x'_1, \ldots, x'_m) \wedge (x_1, \ldots, x_m) \in \mathcal{D}_{f_k})$$

Továbbá minden $i \in [1..k-1]$ esetén jelölje Q_i az i-edik program Q_{i+1}-hez tartozó leggyengébb előfeltételét. Könnyen látható, hogy az értékadás leggyengébb előfeltételére vonatkozó szabályt alkalmazva:

$$Q_1 : (Q \wedge (f_1(x_1, \ldots, x_m), \ldots, f_k(x_1, \ldots, x_m)) = (f_1, \ldots, f_k)(x'_1, \ldots, x'_m) \wedge \\ (x_1, \ldots, x_m) \in \mathcal{D}_{f_1} \wedge \cdots \wedge (x_1, \ldots, x_m) \in \mathcal{D}_{f_k})$$

Ha most Q-t és Q_1-et összehasonlítjuk, észrevehetjük, hogy megegyeznek. A szekvencia levezetési szabálya és a specifikáció tétele alapján a

$y_{1_1}, \ldots, y_{1_{n_1}} := f_1(x_1, \ldots, x_m)$
$\vdots$
$y_{k_1}, \ldots, y_{k_{n_k}} := f_k(x_1, \ldots, x_m)$

program megoldása a fent specifikált feladatnak, azaz kiszámítja $(f_1, \ldots, f_k)$-t.

Rekurzió. Legyen n rögzített, $f \in \mathbb{N}^n \to \mathbb{N}$ és $g \in \mathbb{N}^{n+2} \to \mathbb{N}$. Tegyük fel, hogy mindketten kiszámíthatók jól konstruált programokkal vagy egyszerű értékadásokkal. Az f függvény g szerinti rekurziója által meghatározott feladat specifikációja:

$$\begin{array}{lcccccc} A = & \mathbb{N} & \times \ldots \times & \mathbb{N} & \times & \mathbb{N} \\ & x_1 & & x_{n+1} & & y \\ B = & \mathbb{N} & \times \ldots \times & \mathbb{N} & & \\ & x'_1 & & x'_{n+1} & & \end{array}$$

$$Q : (x_1 = x'_1 \wedge \cdots \wedge x_{n+1} = x'_{n+1} \wedge (x_1, \ldots, x_{n+1}) \in \mathcal{D}_{\varrho(f,g)})$$

$$R : (Q \wedge y = \varrho(f,g)(x'_1, \ldots, x'_{n+1}))$$

Jelölje az f-et és a g-t kiszámító programot $y := f(x_1, \ldots, x_n)$ illetve $y := g(x_1, \ldots, x_{n+2})$. Oldjuk meg a feladatot egy olyan ciklussal, amelynek invariáns tulajdonsága:

$$P : (Q \wedge k \in [1..x_{n+1}] \wedge y = \varrho(f,g)(x'_1, \ldots, x_n, k))$$

A ciklus levezetési szabályát vizsgálva azt találjuk, hogy Q-ból nem következik P. Ezért adunk egy olyan Q' feltételt, amelyből már következik P, és adunk egy programot, amely Q-ból Q'-be jut (így a megoldóprogram egy szekvencia lesz, amelynek második része egy ciklus). Legyen Q' az alábbi:

$$P : (Q \wedge k = 1 \wedge y = f(x_1, \ldots, x_n))$$

Ez a $k, y := 1, f(x_1, \ldots, x_n)$ szimultán értékadással elérhető. Az értékadás leggyengébb előfeltételére vonatkozó szabály felhasználásával könnyen látható, hogy az következik Q-ból.

A ciklus levezetési szabályának második pontja alapján a ciklusfeltétel $k \neq x_{n+1}$ lesz.

A harmadik pontnak megfelelően válasszuk a $t = x_{n+1} - k$ kifejezést terminálló függvénynek.

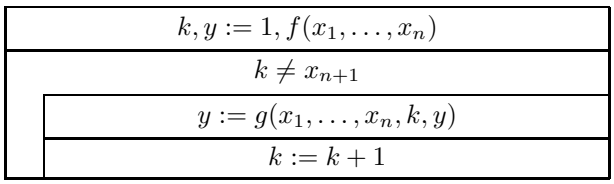
Az ötödik pont azt írja le, hogy az imént definiált terminálló fügvény értékének a ciklusmagban csökkennie kell. Ez elérhető a k eggyel való növelésével.

A négyes pont kielégítéséhez vizsgáljuk meg, hogy mi lesz a leggyengébb előfeltétele a k-t növelő értékadásnak a P-re vonatkozóan.

$$\begin{array}{rcl} Q'' = lf(k := k+1, P) & = & (Q \wedge k+1 \in [1..x_{n+1}] \ \wedge \\ & & \quad y = \varrho(f,g)(x'_1, \ldots, x'_n, k+1)). \end{array}$$

Most már – a szekvencia levezetési szabálya alapján – csak egy olyan programot kell találnunk, amelyre $P \wedge (k \neq x_{n+1}) \Rightarrow lf(S, Q'')$. Ez a program a rekurzió definíciójához illeszkedően épp $y := g(x_1, \ldots, x_n, k, y)$ lesz.

Így a szekvencia és a ciklus levezetés szabálya valamint a specifikáció tétele garantálja, hogy a



program megoldja a $\varrho(f, g)$ által meghatározott feladatot, azaz kiszámítja f g szerinti rekurzióját.

μ-operátor. Legyen $f \in \mathbb{N}^{n+1} \to \mathbb{N}$, és tegyük fel, hogy f kiszámítható egyszerű értékadással vagy jól konstruált programmal. Tekintsük a $\mu(f)$ által meghatározott feladatot:

$$\begin{array}{lll} A = & \mathbb{N} \times \ldots \times \mathbb{N} & \times \mathbb{N} \\ & x_1 \qquad\quad x_n & \quad y \\ B = & \mathbb{N} \times \ldots \times \mathbb{N} & \\ & x'_1 \qquad\quad x'_n & \end{array}$$

$$Q : (x_1 = x'_1 \wedge \cdots \wedge x_n = x'_n \wedge (x_1, \ldots, x_{n+1}) \in \mathcal{D}_{\mu(f)})$$

$$R : (Q \wedge y = \mu(f)(x'_1, \ldots, x'_n))$$

Jelölje $z := f(x_1, \ldots, x_n, x_{n+1})$ az f-et kiszámító programot. Oldjuk meg a feladatot egy olyan ciklussal, melynek invariánsa:

$$P : (Q \wedge z = f(x_1, \ldots, x_n, y) \wedge \forall i \in [1..y-1] : f(x_1, \ldots, x_n, i) \neq 1)$$

Az invariáns a $z, y := f(x_1, \ldots, x_n, 1), 1$ szimultán értékadással teljesíthető. Könnyen látható, hogy ennek a programnak a P-re vonatkozó leggyengébb előfeltétele

$$\begin{array}{rcl} lf(z, y := f(x_1, \ldots, x_n, 1), 1; P) & = & (Q \wedge f(x_1, \ldots, x_n, 1) = f(x_1, \ldots, x_n, 1) \\ & & \wedge \forall i \in [1..1-1] : f(x_1, \ldots, x_n, i) \neq 1) \end{array}$$

következik Q-ból. A ciklus levezetési szabályának második pontja alapján a ciklusfeltétel $z \neq 1$ lesz.

A feladat előfeltétele garantálja, hogy van olyan $m \in \mathbb{N}$ szám, amelyre $f(x_1, \ldots, x_n, m) = 1$ fennáll. Legyen N egy ilyen tulajdonságú, rögzített természetes szám. Ennek az értéknek a segítségével definiálhatjuk a terminálló függvényt: $t = N - y$. Ez kielégíti a levezetés szabály harmadik pontját.

Az ötödik pont megkívánja, hogy a terminálló függvény a ciklusmag lefutása során csökkenjen. Ezt elérhetjük y eggyel való növelésével.

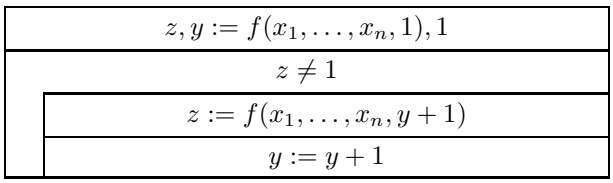
A negyedik pont teljesítéséhez legyen Q' ennek a növelésnek a P-re vonatkozó leggyengébb előfeltétele:

$$Q' : (Q \wedge z = f(x_1, \ldots, x_n, y+1) \wedge \forall i \in [1..y-1] : f(x_1, \ldots, x_n, i) \neq 1)$$

Most már csak találnunk kell egy programot a $P \wedge (z \neq 1)$ és Q' állítások közé. A $z := f(x_1, \ldots, x_n, y+1)$ értékadásra teljesül, hogy

$$P \wedge (z \neq 1) \Rightarrow lf(z := f(x_1, \ldots, x_n, y+1), Q').$$

A specifikáció tétele, valamint a ciklus és a szekvencia levezetési szabálya garantálja, hogy a



program megoldja a $\mu(f)$ által meghatározott feladatot, azaz kiszámítja $\mu(f)$-et.

7.6.3. Következmény

Az előzőekben megmutattuk, hogy ha az alapfüggvények kiszámíthatók egyszerű értékadással, akkor a belőlük – a parciális rekurzív függvényeknél megengedett – operátorokkal felépített függvények kiszámíthatók jól konstruált programokkal. A Church tézis szerint a kiszámítható függvények halmaza megegyezik a parciális rekurzív függvények halmazával. Ezek alapján kimondhatjuk az alábbi tételt:

7.16. TÉTEL: STRUKTURÁLT PROGRAMOZÁS ÉS KISZÁMÍTHATÓSÁG
Minden kiszámítható függvény kiszámítható egy jól konstruált programmal.

7.6.4. Relációk

Eljutván az előző tételhez, fordítsuk most fi gyelmünket a relációk felé. Ahhoz, hogy a relációk kiszámíthatóságát megvizsgálhassuk, defi niálnunk kell a kiszámítható reláció fogalmát.

7.5. Definíció (REKURZÍVAN FELSOROLHATÓ RELÁCIÓ). *Legyen $R \subseteq \mathbb{N}^k \times \mathbb{N}^k$ tetszőleges reláció. R akkor és csak akkor rekurzívan felsorolható, ha van olyan $f \in \mathbb{N}^{2k} \to \mathbb{N}$ parciális rekurzív függvény, amelynek értelmezési tartományára: $\mathcal{D}_f = R$.*

A kiszámíthatóság-elmélethez igazodva, a továbbiakban csak rekurzívan felsorolható relációkkal fogunk foglalkozni. Ahhoz, hogy megmutassuk, hogy minden kiszámítható (rekurzívan felsorolható) feladat – emlékezzünk, hogy minden feladat egy reláció – megoldható strukturált programmal, először megadjuk a rekurzívan felsorolható relációk egy másik jellemzését.

7.17. TÉTEL: KLEENE[1936]
Ha R egy tetszőleges reláció, akkor az alábbi három állítás ekvivalens:

(1) R rekurzívan felsorolható

(2) R egy f parciális rekurzív függvény értékkészlete

(3) $R = \emptyset$ vagy R egy φ rekurzív függvény értékkészlete.

Ennek a tételnek a bizonyítása lásd Ref??? konstruktív, azaz megadja mind f, mind pedig φ felépítését. Mi ezt a φ függvényt fogjuk használni a kiszámítható feladatunk megoldóprogramjában.

Legyen $F \subseteq \mathbb{N}^k \times \mathbb{N}^k$ egy rekurzívan felsorolható reláció, és jelölje φ az előző tétel konstrukciójával kapott (totális) rekurzív függvényt. Specifi káljuk a feladatot az alábbi módon:

$$\begin{array}{lll} A = & \mathbb{N}^k & \times \mathbb{N}^k \\ & x & y \\ B = & \mathbb{N}^k & \\ & x' & \end{array}$$

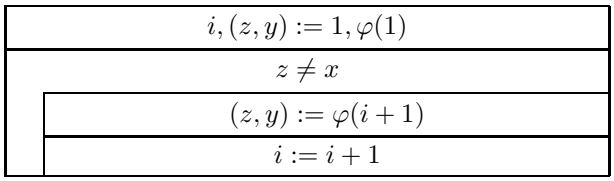
Q : $(x = x' \wedge x \in \mathcal{D}_F)$

R : $(Q \wedge (x, y) \in F)$

Ez a feladat megoldható egy olyan ciklussal, amelynek invariáns tulajdonsága:

P : $(Q \wedge i \in \mathbb{N} \wedge (z, y) = \varphi(i))$

A ciklus levezetési szabályának felhasználásával könnyen belátható, hogy az alábbi program megoldása a fenti feladatnak:



A bizonyításban a terminálló függvény megadásánál ugyanazt a technikát alkalmazzuk, mint a μ-operátornál.

Ezzel az előzőleg függvényekre kimondott tételünket általánosíthatjuk relációkra:

7.18. TÉTEL: STRUKTURÁLT PROGRAMOZÁS ÉS KISZÁMÍTHATÓ RELÁCIÓK
Minden kiszámítható reláció kiszámítható egy jól konstruált programmal.

Megjegyzés. Az egyetlen feltevés, amit a fenti meggondolásokban használtunk az volt, hogy az alapfüggvények kiszámíthatóak. Így aztán ezek az eredmények kiterjeszthetők a relatíve kiszámítható függvények (ugyanezen operátorokkal egy tetszőleges alaphalmazból képzett függvények), vagy a parciális rekurzív funkcionálok halmazára is (Ref[1, page 174]).

8. fejezet

Típuskonstrukciók

Az előző fejezetben megvizsgáltuk, hogy milyen lehetőségeink vannak meglevő programokból újak készítésére. A továbbiakban azt fogjuk megvizsgálni, hogyan használhatunk fel meglévő típusokat új típusok létrehozására. Ezeket a módszereket típuskonstrukciós módszereknek, az általuk megkapható típusokat típuskonstrukcióknak nevezzük.

8.1. A megengedett konstrukciók

Természetesen sokféle lehetőségünk van meglevő típusból újat csinálni, de mi a továbbiakban csak három speciális típuskonstrukcióval fogunk foglalkozni: a direktszorzattal, az unióval és az iterálttal. Ezeket fogjuk megengedett típuskonstrukcióknak nevezni.

Az első típuskonstrukciós módszer, amivel megismerkedünk a direktszorzat. Legyenek $\mathcal{T}_i = (\varrho_i, I_i, \mathbb{S}_i)$ $(i = 1, 2, \ldots, n)$ típusok, és jelöljük $T_1, T_2, \ldots, T_n$-nel a nekik megfelelő típusértékhalmazokat, és $E_1, E_2, \ldots, E_n$-nel pedig a hozzájuk tartozó elemi típusértékhalmazokat, és vezessük be az $E = E_1 \cup E_2 \cup \cdots \cup E_n$ és $B = T_1 \times T_2 \times \cdots \times T_n$ jelölést.

8.1. Definíció (DIREKTSZORZAT). *A $\mathcal{T} = (\varrho, I, \mathbb{S})$ típus direktszorzata a $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n$ típusoknak, ha*

$$\varrho = \varphi_D \circ \psi_D$$

ahol $\varphi_D \subseteq B \times T$, $\psi_D \subseteq E^* \times B$ és

$$\psi_D = \{(\varepsilon, b) \in E^* \times B \mid \forall i \in [1..n] : \exists \varepsilon_i \in E_i^* : (\varepsilon_i, b_i) \in \varrho_i \ \wedge \\ \varepsilon = con(\varepsilon_1, \ldots, \varepsilon_n i)\}.$$

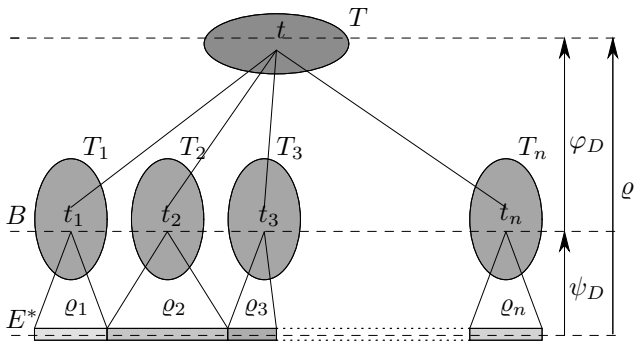
A direktszorzat értékhalmazára bevezetjük a $T = (T_1, T_2, \ldots, T_n)$ jelölést.

Ha a φ_D leképezés kölcsönösen egyértelmű, akkor a direktszorzatot *rekord* típusnak nevezzük. A direktszorzat típusok általában rekordok, de nem mindig. Például tekintsük a racionális számok halmazának egy lehetséges reprezentációját:

$B = \mathbb{Z} \times \mathbb{Z}$, $\varphi_D \subseteq B \times \mathbb{Q}$:

$$((x, y), t) \in \varphi_D \Longleftrightarrow y \neq 0 \wedge t = x/y$$

Egyszerűen látható, hogy a fent definiált φ_D reláció a racionális számok halmazát reprezentálja, de nem kölcsönösen egyértelmű.



8.1. ábra. Rekord konstrukció

Nagyon fontos továbbá, hogy az új típusértékhalmazt (T) ne keverjük össze a közbülső direktszorzattal (B), hiszen egy adott B és T között nagyon sokféle φ_D leképezés megadható, és az új típus szempontjából egyáltalán nem mindegy, hogy ezek közül melyiket választjuk.

Tekintsük például a komplex egészek ($a + bi, a, b \in \mathbb{Z}$ alakú számok) halmazát. Legyen továbbá $B = \mathbb{Z} \times \mathbb{Z}, x, y \in \mathbb{Z}$, és

$$\begin{aligned} \varphi_{D_1}((x,y)) &= x + yi \\ \varphi_{D_2}((x,y)) &= y + xi. \end{aligned}$$

A két φ_D közötti különbség elsősorban akkor válik szignifikánssá, ha például a komplex egészek közötti szorzásműveletet kell implementálnunk a számpárok szintjén, hiszen ekkor az első és a második komponens értékét az

$$(a + bi)(c + di) = (ac - bd) + (ad + bc)i$$

formula alapján különböző módon kell kiszámítani.

A következő módszer, amivel régi típusokból újakat hozhatunk létre, az unió. Legyenek $\mathcal{T}_i = (\varrho_i, I_i, \mathbb{S}_i)$ $(i = 1, 2, \ldots, n)$ típusok, és jelölje $T_1, T_2, \ldots, T_n$ a hozzájuk tartozó típusértékhalmazokat, illetve $E_1, E_2, \ldots, E_n$ a nekik megfelelő elemi típusértékhalmazokat. Vezessük be továbbá az $E = E_1 \cup E_2 \cup \cdots \cup E_n$ és $B = T_1 \cup T_2 \cup \cdots \cup T_n$ jelöléseket.

8.2. Definíció (UNIÓ). *Azt mondjuk, hogy a $\mathcal{T} = (\varrho, I, \mathbb{S})$ típus uniója a $\mathcal{T}_1$, $\mathcal{T}_2$, …, $\mathcal{T}_n$ típusoknak, ha*

$$\varrho = \varphi_U \circ \psi_U$$

ahol $\varphi_U \subseteq B \times T$, $\psi_U \subseteq E^* \times B$ és

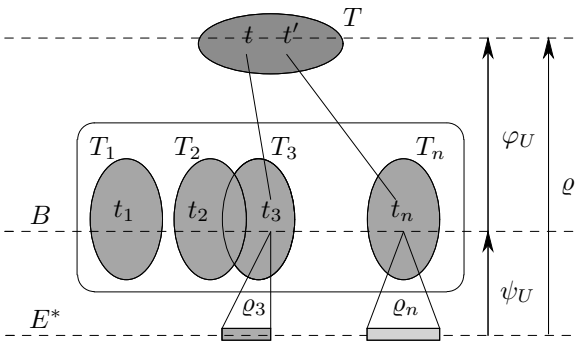
$$\psi_U = \{(\varepsilon, b) \in E^* \times B \mid \exists i \in [1..n] : (\varepsilon, b) \in \varrho_i\}$$

Az unió értékhalmazának jelölése: $T = (T_1; T_2; \ldots; T_n)$.

Itt is külön elnevezést adtunk annak az esetnek, amikor a φ_U leképezés kölcsönösen egyértelmű, ekkor az uniót *egyesítésnek* nevezzük.

Ebben az esetben is nagyon fontos, hogy mindig megkülönböztessük a konstrukció közbülső szintjén levő uniót (B) az új típusértékhalmaztól (T).

A harmadik megengedett típuskonstrukciós művelet az iterált, amellyel egy meglevő típusból alkothatunk új típust. Legyen $\mathcal{T}_0 = (\varrho_0, I_0, \mathbb{S}_0)$ típus, T_0 a neki megfelelő típusértékhalmaz, és E a $\mathcal{T}_0$ típus elemi típusértékhalmaza.



8.2. ábra. Unió konstrukció

8.3. Definíció (ITERÁLT). *Azt mondjuk, hogy a* $\mathcal{T} = (\varrho, I, \mathbb{S})$ *típus iteráltja a* $\mathcal{T}_0$ *típusnak, ha*

$$\varrho = \varphi_I \circ \psi_I$$

ahol $\varphi_I \subseteq B \times T$, $\psi_I \subseteq E^* \times T_0^*$ *és*

$$\begin{aligned} \psi_I \quad = \quad & \{(\varepsilon, b) \in E^* \times T_0^* \mid \exists \varepsilon_1, \ldots, \varepsilon_{|b|} \in E^* : (\varepsilon_i, b_i) \in \varrho_0 \ \wedge \\ & \varepsilon = kon(\varepsilon_1, \ldots, \varepsilon_{|b|})\} \end{aligned}$$

Az iterált típusértékhalmazának jelölése: $T = it(T_0)$.

Az iterált típuskonstrukciónak három speciális esetét különböztetjük meg aszerint, hogy a φ_I leképezésre milyen feltételek teljesülnek.

- Ha a φ_I leképezés kölcsönösen egyértelmű, akkor *sorozat* típuskonstrukcióról beszélünk, és típusértékhalmazát $T = seq(T_0)$-lal jelöljük.

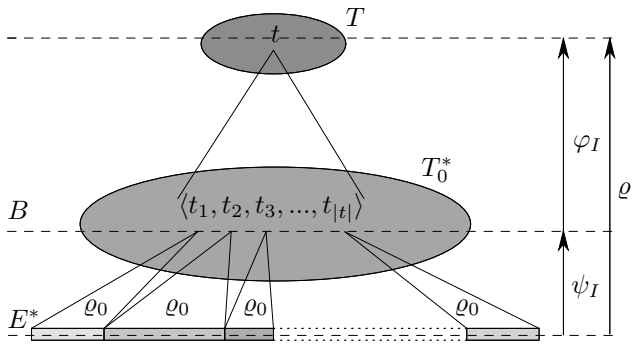
- Ha
$$(\alpha, t), (\beta, t) \in \varphi_I \Leftrightarrow \alpha \in perm(\beta),$$
akkor az iterált konstrukcót *kombináció* típusnak nevezzük. A kombináció értékhalmazának jelölése: $T = com(T_0)$.

- Ha
$$(\alpha, t), (\beta, t) \in \varphi_I \Leftrightarrow \bigcup_{i=1}^{|\alpha|} \{\alpha_i\} = \bigcup_{i=1}^{|\beta|} \{\beta_i\},$$
akkor *halmaz* típuskonstrukcióról beszélünk. A halmaz típus értékhalmazának jelölése: $T = set(T_0)$.

Természetesen az imént felsorolt három eset csak speciális formája az iteráltképzésnek; létezik olyan iterált is, amely egyik fenti kritériumot sem teljesíti.

8.2. Szelektorfüggvények

Az előzőkben defi niált típuskonstrukciókra most bevezetünk néhány olyan függvényt és jelölést, amelyek leegyszerűsítik a rájuk vonatkozó állítások, programok megfogalmazását.



8.3. ábra. Iterált konstrukció

Legyen $T = (T_1, T_2, \ldots, T_n)$. A $\varphi_D^{(-1)}$ függvény komponenseit a T rekord szelektorfüggvényeinek, vagy röviden szelektorainak nevezzük.

A fenti rekordnak tehát pontosan n darab szelektora van, és ha s_i-vel jelöljük az i-edik szelektort, akkor $s_i : T \to T_i$, és

$$\forall t \in T : \varphi_D(s_1(t), s_2(t), \ldots, s_n(t)) = t.$$

Tehát a szelektorfüggvényeket arra használhatjuk, hogy lekérdezzük a rekord egyes mezőinek (komponenseinek) az értékét. A szelektorokat bele szoktuk írni a típusértékhalmaz jelölésébe; a fenti esetben a szelektorokkal felírt jelölés:

$$T = (s_1 : T_1, s_2 : T_2, \ldots, s_n : T_n).$$

A rekordtípushoz hasonlóan az egyesítéshez is bevezetünk szelektorfüggvényeket. Mivel az unió esetében a közbülső szinten a típusértékhalmazok uniója szerepel, így nincs értelme komponensről beszélni. Hogyan definiáljuk ez esetben a szelektorokat? Azt fogják visszaadni, hogy egy adott T-beli elemet melyik eredeti típusértékhalmaz egy eleméhez rendelte hozzá a φ_U függvény.

Legyen $T = (T_1; T_2; \ldots; T_n)$ egyesítés típus, $s_i : T \to \mathbb{L}$ $(i = 1, \ldots, n)$. Azt mondjuk, hogy az s_i logikai függvények a T egyesítés szelektorai, ha $\forall i \in [1..n]$: $\forall t \in T$:

$$s_i(t) = \left(\varphi_U^{(-1)}(t) \in T_i\right).$$

A rekordtípushoz hasonló módon az egyesítés szelektorait is bele szoktuk írni az új típusértékhalmaz jelölésébe. A szelektorokkal felírt típusértékhalmaz jelölése:

$$T = (s_1 : T_1; s_2 : T_2; \ldots; s_n : T_n).$$

Az iterált típuskonstrukciók közül a sorozathoz definiálunk szelektorfüggvényt. A sorozattípusban a közbülső szinten T_0-beli sorozat szerepel, a szelektor ennek a sorozatnak a tagjait adja vissza.

Formálisan: Legyen $T = seq(T_0)$. Az $s : T \times \mathbb{N} \to T_0$ parciális függvény a T szelektorfüggvénye, ha $\forall t \in T : \forall i \in [1..|\varphi_I^{(-1)}(t)|]$:

$$s(t, i) = \varphi_I^{(-1)}(t)_i.$$

A sorozat szelektorát nem szoktuk külön elnevezni, helyette indexelést alkalmazunk, azaz az $t_i = s(t, i)$ jelölést használjuk.

8.3. Az iterált specifikációs függvényei

Ha az iterált típus az előzőekben bevezetett három speciális osztály valamelyikébe tartozik, akkor további függvényeket definiálunk hozzá.

Legyen $T = it(T_0)$, $(\alpha, t) \in \varphi_i$, és tegyük fel, hogy az iterált sorozat, kombináció, vagy halmaz. Ekkor $dom : T \to \mathbb{N}_0$,

$$dom(t) = \begin{cases} |\alpha|, & \text{ha } T = seq(T_0) \text{ vagy } T = com(T_0) \\ |\bigcup_{i=1}^{|\alpha|} \{\alpha_i\}|, & \text{ha } T = set(T_0) \end{cases}$$

A dom függvény tehát a t elemeinek számát adja meg. A függvény jóldefiniált, ugyanis felhasználva a sorozat, kombináció és halmaz típus definícióját, könnyen látható, hogy a függvényérték független az α választásától.

A továbbiakban a sorozattípussal fogunk foglalkozni. Ahol külön nem jelöljük, ott $T = seq(T_0)$, $(\alpha, t) \in \varphi_I$, $\alpha = \langle \alpha_1, \alpha 2, \ldots \alpha_{|\alpha|} \rangle$.

- Nem üres sorozat első és utolsó eleme: $lov : T \to T_0$, $hiv : T \to T_0$,

$$\begin{aligned} lov(t) &= \alpha_1 \\ hiv(t) &= \alpha_{|\alpha|} \end{aligned}$$

- Sorozat kiterjesztése a sorozat elején, vagy végén (legyen $e \in T_0$): $loext : T \times T_0 \to T$, $hiext : T \times T_0 \to T$,

$$\begin{aligned} loext(t, e) &= \varphi_I(con(\langle e \rangle, \alpha)) \\ hiext(t, e) &= \varphi_I(con(\alpha, \langle e \rangle)) \end{aligned}$$

- Nem üres sorozat első, vagy utolsó elemének elhagyásával kapott sorozat: $lorem : T \to T$, $hirem : T \to T$,

$$\begin{aligned} lorem(t) &= \varphi_I(\langle \alpha_2, \ldots, \alpha_{|\alpha|} \rangle) \\ hirem(t) &= \varphi_I(\langle \alpha_1, \ldots, \alpha_{|\alpha|-1} \rangle) \end{aligned}$$

8.4. A függvénytípus

A gyakorlatban nagyon fontos szerepet játszik egy speciális rekordtípus. Legyen H egy tetszőleges (megszámlálható) halmaz, amelyen van egy rákövetkezési reláció. Jelöljük ezt a rákövetkezési relációt $succ$-cal, és inverzére vezessük be a $pred$ jelölést.

8.4. Definíció (FÜGGVÉNY TÍPUS). *Legyen E egy tetszőleges típus értékhalmaza. Ekkor az $F = (H, seq(E))$ rekordot függvénytípusnak nevezzük, és $F = fun(H, E)$-vel jelöljük.*

A függvénytípusra is bevezetünk néhány fontos specifikációs függvényt. A továbbiakban legyen $F = fun(H, E)$, $((h, t), f) \in \varphi_D$. Ekkor

- $dom : F \to \mathbb{N}_0$,

$$dom(f) = dom(t)$$

- $lob : F \to H$,

$$lob(f) = h$$

- $hib : F \to H$,
$$hib(f) = succ^{dom(f)-1}(h)$$
- $lov : F \to E$,
$$lov(f) = lov(t)$$
- $hiv : F \to E$,
$$hiv(f) = hiv(t)$$
- $loext, hiext : F \times E \to F$,
$$\begin{aligned} loext(f,e) &= \varphi_D(pred(h), loext(t,e)) \\ hiext(f,e) &= \varphi_D(h, hiext(t,e)) \end{aligned}$$
- $lorem, hirem : F \to F$,
$$\begin{aligned} lorem(f) &= \varphi_D(succ(h), lorem(t)) \\ hirem(f) &= \varphi_D(h, hirem(t)) \end{aligned}$$

A sorozathoz hasonlóan a függvénytípusra is bevezetünk egy szelekciós parciális függvényt. TekintsÜk a fentiekben használt f-et. Ekkor $s_f : H \to E$, $\mathcal{D}_{s_f} = \{succ^i(lob(f)) \mid 0 \leq i < dom(f)\}$, és ha $g \in \mathcal{D}_{s_f}$, $g = succ^k(lob(f))$, akkor:

$$s_f(g) = t_{k+1}$$

A függvénytípus szelektorfüggvényét nem szoktuk külön elnevezni, helyette a matematikában – a függvény helyettesítési értékének jelölésére – használt zárójeles hivatkozást használjuk, azaz

$$f(g) := s_f(g).$$

A függvénytípus elnevezés azt a szemléletes képet tükrözi, hogy egy függvénytípusú érték felfogható egy $H \to E$ típusú parciális függvénynek, amelynek értelmezési tartománya a "lob-tól a hib-ig tart", és értékeit pedig a sorozat komponens tartalmazza.

Az előbbiekben bevezetett dom, lov, hiv, lob, hib függvényeket kiterjeszthetjük az egész állapottérre is: komponáljuk a megfelelő változóval. Tehát ha például x egy sorozattípusú változó, akkor $dom \circ x$ egy az egész állapottéren értelmezett függvény. Az ilyenfajta függvénykompozíciókra bevezetünk egy újabb jelölést: ha t a fenti függvények valamelyike, és x a neki megfelelő típusú változó, akkor az $t \circ x$ helyett $x.t$-t írunk.

8.5. A típuskonstrukciók típusműveletei

A típuskonstrukciók eddigi tárgyalásából még hiányzik valami: nem beszéltünk még a konstruált típusok műveleteiről. Az előbb felsorolt speciális esetekhez – az imént defi niált függvények segítségével – bevezetünk néhány típusműveletet.

A továbbiakban megengedett feltételnek fogjuk nevezni azokat az $A \to \mathbb{L}$ állításokat, amelyek lehetnek elágazás, vagy ciklus feltételei.

Legyen $T = (s_1 : T_1, \ldots, s_n : T_n)$ rekord, $t : T$, $t_i : T_i$ ($i \in [1..n]$). Mivel t az állapottér változója, t komponálható a szelektorfüggvényekkel, és így az állapotéren értelmezett függvényeket kapunk. A $s_i \circ t$ függvénykompozíciót a továbbiakban $t.s_i$-vel

fogjuk jelölni. Egy rekord típusnál a szelektorfüggvény használatát megengedettnek tekintjük.

Ezen kívül bevezetjük a $t.s_i := t_i$ jelölést is. Ezen azt a $t := t'$ értékadást értjük, amelyben $t'.s_i = t_i$ és t' minden más komponense megegyezik t megfelelő komponensével.

A fenti típusműveletek arra adnak lehetőséget, hogy egy rekord „mezőinek" az értékét lekérdezhessük, illetve megváltoztathassuk. A fent definiált műveletben zavaró lehet, hogy egy függvény helyettesítési értékének ($t.s_i$) „adunk értéket". Ezért fontos megjegyezni, hogy ez csupán egy jelölése az értékadásnak.

Legyen $T = (s_1 : T_1; \ldots; s_n : T_n)$ egyesítés, $t : T$, $t_i : T_i$ ($i \in [1..n]$). Ekkor a rekord típusnál bevezetett jelölést az egyesítés esetén is bevezetjük, $t.s_i$-n, az $s_i \circ t$ kompozíciót értjük, és megengedett függvénynek tekintjük.

Ezen kívül megengedett művelet a $t := \varphi_U(t_i)$ értékadás. Ennek az értékadásnak a jelölését leegyszerűsítjük, a továbbiakban $t := t_i$ alakban fogjuk használni.

A fenti értékadást bizonyos ésszerű korlátozások bevezetésével „megfordíthatjuk". Így kapjuk az alábbi parciális értékadást: $t_i := t$. Ez az értékadás csak akkor végezhető el, ha $t.s_i$ igaz.

A sorozat típuskonstrukció nagyon gyakori, és sokféle művelet definiálható vele kapcsolatban. Attól függően, hogy melyeket tekintjük megvalósítottnak, különböző konstrukciókról beszélünk. Most előbb megadunk néhány lehetséges műveletet, majd ezután a sorozat típusokat osztályba soroljuk a megengedett műveleteik alapján.

Legyen a továbbiakban $T = seq(E)$, $t : T$, $e : E$. Ekkor az iménti szelektorokhoz hasonlóan bevezetjük az alábbi jelöléseket:

$$\begin{aligned} dom \circ t &\rightarrow t.dom \\ lov \circ t &\rightarrow t.lov \\ hiv \circ t &\rightarrow t.hiv \end{aligned}$$

Természetesen $t.lov$ és $t.hiv$ csak parciális függvények. Ezen kívül az alábbi (esetleg parciális) értékadásokra a következő jelöléseket fogjuk használni:

$$\begin{aligned} t := lorem(t) &\rightarrow t : lorem \\ t := hirem(t) &\rightarrow t : hirem \\ t := loext(t, e) &\rightarrow t : loext(e) \\ t := hiext(t, e) &\rightarrow t : hiext(e) \\ e, t := lov(t), lorem(t) &\rightarrow e, t : lopop \\ e, t := hiv(t), hirem(t) &\rightarrow e, t : hipop \end{aligned}$$

A bevezetett jelölések első látásra zavarba ejtőnek tűnhetnek, hiszen ugyanazt a kulcsszót a baloldalon függvényként, a jobboldalon pedig a művelet neveként használjuk. Lényeges ezért megjegyezni, hogy a jobb oldalon található műveletek csa a baloldali értékadás egyszerűsítő *jelölései*.

Attól függően, hogy a fent definiált műveletek közül melyeket tekintjük megengedettnek, különböző konstrukciókról beszélünk.

8.5. Definíció (SPECIÁLIS SOROZATTÍPUSOK). *Legyen $T = seq(E)$. Ekkor a T*

- szekvenciális input file, *ha csak a $lopop$ a megengedett művelet,*
- szekvenciális output file, *ha csak a $hiext$ a megengedett művelet,*
- verem, *ha a megengedett műveletek a $loext$ és a $lopop$, vagy a $hiext$ és a $hipop$,*
- sor, *ha a megengedett műveletek a $hiext$ és a $lopop$, vagy a $loext$ és a $hipop$.*

Ahhoz, hogy a szekvenciális input file a $lopop$ művelettel használható legyen, tudnunk kell, hogy mikor olvastuk ki az utolsó elemet a file-ból. Ezt a problémát úgy szoktuk megoldani, hogy bevezetünk egy extremális elemet, és kikötjük, hogy a file-ban ez az utolsó elem (tehát még az üres file is tartalmazza). Ez a technika valósul meg azokban az operációs rendszerekben, ahol a szövegfile-ok végét file-vége (EOF) karakter jelzi.

Mivel a $lopop$ művelet bizonyos esetekben kényelmetlen lehet – gondoljunk arra, amikor nehézkes extremális elemet találni –, bevezetünk egy másik olvasóműveletet is. Használjuk az olvasás sikerességének jelzésére a $\{norm, abnorm\}$ halmaz elemeit. Ekkor az $sx, dx, x : read$ műveleten az alábbi értékadásokat értjük:

$$sx, dx, x : read \rightarrow \begin{cases} sx, dx, x := norm, lov(x), lorem(x), & \text{ha } dom(x) \neq 0 \\ sx := abnorm, & \text{ha } dom(x) = 0 \end{cases}$$

Ha egy szekvenciális file-ra a $read$ művelet van megengedve, akkor nincs szükség extremális elemre, helyette az sx változó értéke alapján lehet eldönteni, hogy végére értünk-e a file-nak.

Legyen $F = fun(H, E)$, $f : F$, $e : E$, $i : H$. Ekkor a sorozat típushoz hasonlóan bevezetjük az alábbi jelöléseket:

$$\begin{aligned} dom \circ f &\rightarrow f.dom \\ lov \circ f &\rightarrow f.lov \\ hiv \circ f &\rightarrow f.hiv \\ lob \circ f &\rightarrow f.lob \\ hib \circ f &\rightarrow f.hib \end{aligned}$$

A fenti függvényeken kívül a függvény típus szelektorfüggvényét $f(i)$-t is megengedettnek tekintjük. A rekord típusnál bevezetett szelektorra (mezőre) vonatkozó értékadásnak jelen esetben is van megfelelője: az $f(i) := e$ parciális értékadás. Az értékadás azért parciális, mert csak akkor végzhető el, ha $f.lob \leq i \leq f.hib$. Ekkor a fenti jelölésen azt az $f := f'$ értékadást értjük, amelyre:

$$\begin{gathered} f'.lob = f.lob \wedge f'.hib = f.hib \wedge f'(i) = e \wedge \\ \forall j \in [f.lob..f.hib] : j \neq i \rightarrow f'(j) = f(j). \end{gathered}$$

A sorozatokra bevezetett kiterjesztő és elhagyó műveleteket függvény típusra is definiáljuk:

$$\begin{aligned} f := lorem(f) &\rightarrow f : lorem \\ f := hirem(f) &\rightarrow f : hirem \\ f := loext(f, e) &\rightarrow f : loext(e) \\ f := hiext(f, e) &\rightarrow f : hiext(e) \end{aligned}$$

Ha ezen utolsó csoportban felsorolt műveleteket egy függvénytípusra nem engedjük meg, akkor egy speciális függvénytípushoz, a vektorhoz jutunk. Az általános függvénytípustól megkülönböztetendő a vektortípusra külön jelölést vezetünk be: $V = vekt(H, E)$. Ha azt is jelölni kívánjuk, hogy mik a vektor indexhatárai, akkor a típust $V = vekt([ah..fh] : E)$-vel jelöljük. További jelölésbeli eltérés az általános függvénytípustól: a szelektorfüggvény jelölésére nem a kerek, hanem a szögletes zárójelet használjuk.

II. rész

Programozási módszertan

9. fejezet

Alapvető programozási tételek

Legelőször – a levezetési szabályok használatát is ismertetve – néhány egyszerű feladatra keresünk megoldóprogramot.

9.1. Összegzés

Legyen adott az $f : \mathbb{Z} \to \mathbb{Z}$ függvény. Feladatunk az, hogy egy adott $[m..n] \subset \mathbb{Z}$ intervallumban összegezzük az f függvény értékeit. Specifikáljuk először a feladatot.

$$\begin{array}{lccccc} A = & \mathbb{Z} & \times & \mathbb{Z} & \times & \mathbb{Z} \\ & m & & n & & s \\ B = & \mathbb{Z} & \times & \mathbb{Z} & & \\ & m' & & n' & & \end{array}$$

$$Q : (m = m' \wedge n = n' \wedge m \leq n + 1)$$

$$R : (Q \wedge s = \sum_{i=m}^{n} f(i))$$

Az előfeltételben szereplő $m \leq n + 1$ feltétel azt fogalmazza meg, hogy az m és n számok egy – legfeljebb üres – intervallumot határoznak meg. (Az intervallum akkor lesz üres, ha a kezdőpontja eggyel nagyobb mint a végpontja.)

Próbáljuk a feladatot ciklussal megoldani: ehhez a ciklus levezetési szabálya alapján keresnünk kell egy invariáns tulajdonságot (P), ami a ciklus futásának egy közbülső állapotát írja le. Fogalmazza meg ez az invariáns azt a tulajdonságot, hogy az összegzést az intervallum egy kezdőszeletére már elvégeztük. Ennek formális leírásához bővítsük ki az állapotteret egy újabb egész típusú komponenssel (k), amely azt fogja mutatni, hogy hol tartunk az összegzésben. Az új állapottér tehát legyen:

$$\begin{array}{lccccccc} A' = & \mathbb{Z} & \times & \mathbb{Z} & \times & \mathbb{Z} & \times & \mathbb{Z} \\ & m & & n & & s & & k \end{array}$$

Ekkor az invariáns tulajdonság:

$$P : (Q \wedge k \in [m - 1..n] \wedge s = \sum_{i=m}^{k} f(i))$$

Vizsgáljuk meg most a ciklus levezetési szabálya feltételeit:

1) $Q \Rightarrow P$: Ez a feltétel jól láthatóan nem teljesül (sőt fordítva áll fenn). Mit lehet ilyenkor tenni? Vagy a Q-t, vagy a P-t meg kell változtatni. A Q azonban a feladatot definiálja, azt nem változtathatjuk meg, a P-t pedig mi választottuk úgy,

hogy egy közbülső állapotot írjon le, ezért azt sem lenne szerencsés eldobni. Van azonban egy harmadik lehetőség is: keressünk egy olyan Q' állítást, amelyre már $Q' \Rightarrow P$ teljesül, és oldjuk meg a feladatot egy olyan szekvenciával, amelynek második része egy olyan ciklus lesz, melynek előfeltétele a Q', az utófeltétele pedig R, első fele pedig egy a Q-ból Q'-be képező program (azaz alkalmazzuk a szekvencia levezetési szabályát). Legyen ez a Q' állítás:

$Q' : (Q \wedge k = m - 1 \wedge s = 0)$

Könnyen látható, hogy ekkor $Q' \Rightarrow P$ fennáll. Most még keresnünk kell egy olyan programot, ami Q-ból Q'-be jut. A $k, s := m - 1, 0$ értékadás megfelel ennek a kritériumnak, hiszen az értékadás leggyengébb előfeltételéről szóló tétel alapján:

$Q \Rightarrow lf(k, s := m - 1, 0, Q') = (Q \wedge m - 1 = m - 1 \wedge 0 = 0) = Q.$

2) $P \wedge \neg\pi \Rightarrow R$: Ebből a feltételből meghatározhatjuk, hogy mi lesz a ciklusunk feltétele. Ehhez azt kell megnézni, hogy mikor következik az invariáns tulajdonságból az utófeltétel. A kettőt összehasonlítva észrevehetjük, hogy ez $k = n$ esetén van így. Tehát

$\neg\pi = (k = n)$, azaz $\pi = (k \neq n)$.

3) $P \wedge \pi \Rightarrow t > 0$: A feltétel teljesítéséhez keresnünk kell egy olyan – az állapottéren értelmezett – egészértékű függvényt, amely az invariáns és a ciklusfeltétel fennállása esetén pozitív. Mivel a változók is az állapottér függvényei, a terminálófüggvényt rajtuk keresztül fogjuk kifejezni. Vegyük észre, hogy ha $k \in [m - 1..n]$ (P miatt) és $k \neq n$ (π miatt), akkor $n - k$ értéke pozitív lesz. Legyen tehát:

$t = n - k$.

5) $P \wedge \pi \wedge t = t_0 \Rightarrow lf(S_0, t < t_0)$: Mivel már van terminálófüggvényünk, vegyük előre a levezetési szabály utolsó feltételét, ami azt kívánja meg, hogy a ciklusmag csökkentse a terminálófüggvényt. Mivel az utófeltétel szerint n értékét meg kell tartanunk, a terminálófüggvényt k növelésével tudjuk csökkenteni. Mennyivel növeljük k-t? Legalább 1-gyel. Ha egy kicsit előre tekintünk, és figyelembe vesszük, hogy a ciklusmagnak az invariánst meg kell tartania (ez a 4. pont), akkor látható, hogy k-t legfeljebb 1-gyel növelhetjük meg. Tehát a $k := k + 1$ értékadás csökkenti a terminálófüggvény értékét, ui:

$P \wedge \pi \wedge n - k = t_0 \Rightarrow lf(k := k + 1, n - k < t_0) = n - (k + 1) < t_0.$

4) $P \wedge \pi \Rightarrow lf(S_0, P)$: Meg kell még vizsgálnunk, hogy a $k := k + 1$ értékadás jó lesz-e ciklusmagnak, azaz megtartja-e az invariánst. Írjuk fel a P-hez tartozó leggyengébb előfeltételét:

$$lf(k := k + 1, P) = (Q \wedge k + 1 \in [m - 1..n] \wedge s = \sum_{i=m}^{k+1} f(i))$$

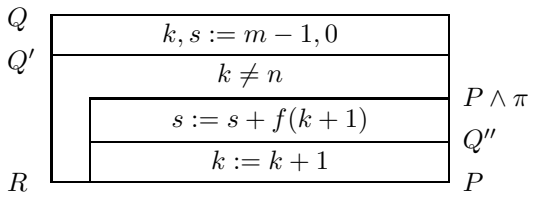
Látható, hogy ez $P \wedge \pi$-ből nem következik. Jelöljük hát a fenti feltételt Q''-vel, és legyen a ciklusmag egy olyan szekvencia, amelynek közbülső feltétele Q'' és második tagja a $k := k+1$ értékadás. Ekkor már nincs más dolgunk, mint találni egy olyan programot, ami $P \wedge \pi$-ből Q''-be képez és t értékét nem változtatja meg. Nézzük meg, hogy mi nem teljesül Q''-ben: mivel $k \in [m - 1..n]$ (P) és $k \neq n$ (π), $k + 1 \in [m - 1..n]$ fennáll. s értéke viszont nem jó, mert P szerint

csak k-ig tartalmazza f értékeinek összegét, Q'' szerint pedig már $k+1$-ig kell. A fenti meggondolás alapján tehát s növelése $f(k+1)$-gyel jó lesz, azaz:

$$P \wedge \pi \Rightarrow lf(s := s + f(k+1), Q'') =$$

$$= (Q \wedge k+1 \in [m-1..n] \wedge s + f(k+1) = \sum_{i=m}^{k+1} f(i)).$$

A fenti levezetés alapján kimondható az alábbi tétel:
Tétel: Az alábbi struktogram formában megadott program megoldása a fent specifikált feladatnak:



Bizonyítás: A tétel a szekvencia és a ciklus levezetési szabályából, a specifikáció tételéből és a fenti levezetésből következik. □

9.2. Számlálás

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

$$\begin{array}{lccccc} A = & \mathbb{Z} & \times & \mathbb{Z} & \times & \mathbb{N}_0 \\ & m & & n & & d \end{array}$$

$$\begin{array}{lccc} B = & \mathbb{Z} & \times & \mathbb{Z} \\ & m' & & n' \end{array}$$

$$Q : (m = m' \wedge n = n' \wedge m \leq n+1)$$

$$R : (Q \wedge d = \sum_{i=m}^{n} \chi(\beta(i)))$$

A fenti specifikációban $\chi : \mathbb{L} \to \{0,1\}$, amelyre $\chi(igaz) = 1$ és $\chi(hamis) = 0$ A feladat megoldása analóg az összegzés tételénél leírtakkal. A ciklus invariáns tulajdonságát itt is az utófeltétel gyengítésével kapjuk (az intervallum egy tetszőleges kezdőszeletére írjuk fel):

$$P : (Q \wedge k \in [m-1..n] \wedge d = \sum_{i=m}^{k} \chi(\beta(i)))$$

A továbbiakban a ciklus levezetési szabályában szereplő feltételekhez címszószerűen felsoroljuk a levezetés során előforduló elemeket.

1) $Q' : (Q \wedge k = m-1 \wedge d = 0)$,

2) $\pi = (k \neq n)$,

3) $t = n - k$,

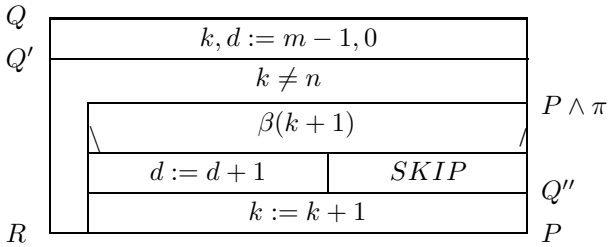
5) a $k := k+1$ értékadás csökkenti a terminálófüggvény értékét,

4) $Q'' = (Q \wedge k{+}1 \in [m{-}1..n] \wedge d = \sum_{i=m}^{k+1} \chi(\beta(i)))$. Itt azt kell megvizsgálni, hogy $P \wedge \pi$-ből következik-e Q''. Vegyük észre, hogy $\neg\beta(k+1)$ esetén következik, míg $\beta(k{+}1)$ esetén – az összegzéshez hasonlóan – meg kell növelnünk d értékét. Ezért a $P \wedge \pi$-ből Q''-be jutó program az $IF(\beta(k{+}1) : d := d{+}1, \neg\beta(k{+}1) : SKIP)$ elágazás lesz, ugyanis az elágazás levezetési szabályát alkalmazva:

$$\begin{aligned} P \wedge \pi \wedge \beta(k+1) &\Rightarrow lf(d := d+1, Q'') \\ P \wedge \pi \wedge \neg\beta(k+1) &\Rightarrow lf(SKIP, Q'') \end{aligned}$$

miatt $P \wedge \pi \Rightarrow lf(IF, Q'')$ teljesül.

A fenti meggondolások alapján nyilvánvaló az alábbi tétel:
Tétel: Az alábbi struktogram formában megadott program megoldása a fent specifikált feladatnak:



Bizonyítás: A tétel a levezetési szabályokból, és a specifikáció tételéből a fenti meggondolások (levezetés) alapján következik. □

9.3. Maximumkeresés

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

$$\begin{array}{lccccc} A = & \mathbb{Z} & \times & \mathbb{Z} \times \mathbb{Z} & \times & \mathcal{H} \\ & m & & n \quad i & & max \end{array}$$

$$\begin{array}{lccc} B = & \mathbb{Z} & \times & \mathbb{Z} \\ & m' & & n' \end{array}$$

$Q : (m = m' \wedge n = n' \wedge m \leq n)$

$R : (Q \wedge i \in [m..n] \wedge max = f(i) \wedge \forall j \in [m..n] : f(j) \leq f(i))$

Vegyük észre, hogy az előbbiekkel ellentétben ebben a specifikációban nem engedtük meg az üres intervallumot. Ennek oka rendkívül egyszerű: üres intervallumon nincs értelme megkérdezni, hogy hol van a maximum. A feladatot ciklussal oldjuk meg, amelynek invariánsa:

$P : (Q \wedge k \in [m..n] \wedge i \in [m..k] \wedge max = f(i) \wedge \forall j \in [m..k] : f(j) \leq f(i))$

A feladatot megoldó program levezetése hasonlatos az előzőhöz, ezért itt is csak címszavakban soroljuk fel a lépéseket:

1) $Q' = (Q \wedge k = m \wedge i = m \wedge max = f(m))$,

2) $\pi = (k \neq n)$,

3) $t = n - k$,

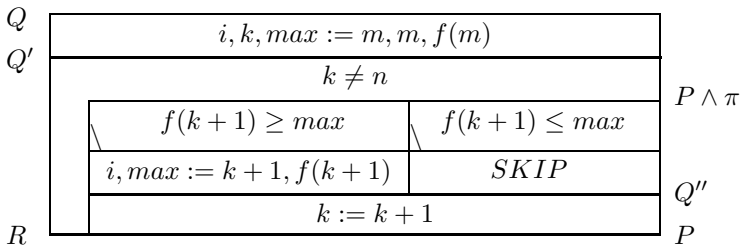
5) a $k := k + 1$ értékadás csökkenti a terminálófüggvény értékét,

4) $Q'' = (Q \wedge k + 1 \in [m..n] \wedge i \in [m..k + 1] \wedge max = f(i) \wedge \forall j \in [m..k + 1] : f(j) \leq f(i))$ A $P \wedge \pi$-ből Q''-be jutó program itt is egy elágazás lesz: $IF(f(k+1) >= max : i, max := k+1, f(k+1), f(k+1) <= max : SKIP)$, ui.

$$\begin{aligned} P \wedge \pi \wedge f(k+1) >= max &\Rightarrow lf(i, max := k+1, f(k+1), Q'') \\ P \wedge \pi \wedge f(k+1) <= max &\Rightarrow lf(SKIP, Q'') \end{aligned}$$

miatt $P \wedge \pi \Rightarrow lf(IF, Q'')$ teljesül.

Tétel: Az alábbi struktogram formában megadott program megoldása a fent specifikált feladatnak:



Bizonyítás: A tétel a levezetési szabályokból, és a specifikáció tételéből a fenti meggondolások (levezetés) alapján következik. □

9.4. Feltételes maximumkeresés

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

$$\begin{array}{lccccccc} A = & \mathbb{Z} & \times & \mathbb{Z} & \times & \mathbb{Z} & \times & \mathcal{H} & \times & \mathbb{L} \\ & m & & n & & i & & max & & l \end{array}$$

$$\begin{array}{lccc} B = & \mathbb{Z} & \times & \mathbb{Z} \\ & m' & & n' \end{array}$$

$Q : (m = m' \wedge n = n' \wedge m \leq n + 1)$

$R : (Q \wedge l = (\exists i \in [m..n] : \beta(i)) \wedge l \rightarrow (i \in [m..n] \wedge \beta(i) \wedge max = f(i) \wedge$
$\forall j \in [m..n] : \beta(j) \rightarrow (f(j) \leq f(i))))$

Vegyük észre, hogy itt újra megengedhető az üres intervallum, és hogy ekkor az a válasz, hogy az intervallumban nincs β tulajdonságú elem. A levezetés az előzőkhöz hasonlóan:

$P : (Q \wedge k \in [m - 1..n] \wedge l = (\exists i \in [m..k] : \beta(i)) \wedge l \rightarrow (i \in [m..k] \wedge \beta(i) \wedge$
$max = f(i) \wedge \forall j \in [m..k] : \beta(j) \rightarrow (f(j) \leq f(i))))$

1) $Q' = (Q \wedge k = m - 1 \wedge l = hamis)$,

2) $\pi = (k \neq n)$,

3) $t = n - k$,

5) a $k := k + 1$ értékadás csökkenti a terminálófüggvény értékét,

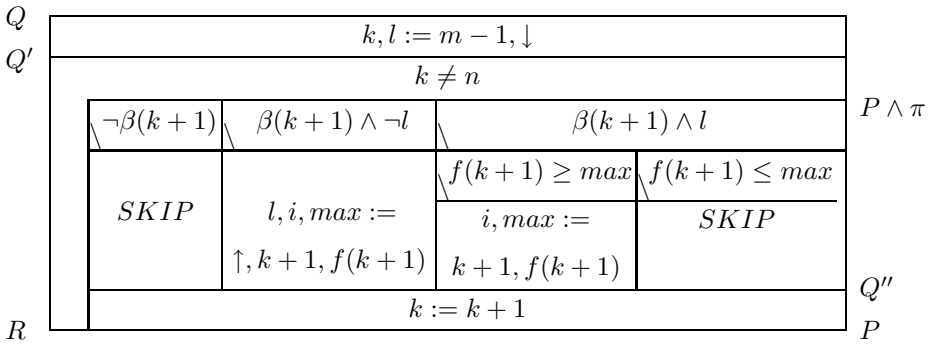
4) Írjuk fel a ciklusmag közbülső feltételét:

Q'' : ($Q \wedge k + 1 \in [m - 1..n] \wedge l = (\exists i \in [m..k + 1] : \beta(i)) \wedge$
$l \rightarrow (i \in [m..k + 1] \wedge \beta(i) \wedge max = f(i) \wedge$
$\forall j \in [m..k + 1] : \beta(j) \rightarrow (f(j) \leq f(i))))$

$P \wedge \pi$ és Q'' összehasonlításával látható, hogy három fő lehetőség van:

- $\neg\beta(k + 1)$: ekkor SKIP,
- $\beta(k+1) \wedge \neg l$: ez az első β tulajdonságú elem, tehát $l, i, max := igaz, k+1, f(k + 1)$,
- $\beta(k + 1) \wedge l$: ekkor a maximumkeresésnél megismert két eset lehetséges:
  - $f(k + 1) >= max$: ekkor $i, max := k + 1, f(k + 1)$,
  - $f(k + 1) <= max$: ekkor $SKIP$.

Tétel: Az alábbi struktogram formában megadott program megoldása a fent specifikált feladatnak:



Bizonyítás: A tétel a levezetési szabályokból, és a specifikáció tételéből a fenti meggondolások (levezetés) alapján következik. □

9.5. Lineáris keresés

Legyen $\beta : \mathbb{Z} \rightarrow \mathbb{L}$ adott tulajdonság. A feladat az, hogy keressük meg azt a legkisebb β tulajdonságú egész számot, amely nem kisebb, mint az adott $m \in \mathbb{Z}$ szám.

$A = \underset{m}{\mathbb{Z}} \times \underset{i}{\mathbb{Z}}$

$B = \underset{m'}{\mathbb{Z}}$

Q : ($m = m' \wedge \exists j \geq m : \beta(j)$)

R : ($Q \wedge i \geq m \wedge \beta(i) \wedge \forall j \in [m..i - 1] : \neg\beta(j)$)

A feladatot ciklussal oldjuk meg. Az invariánshoz gyengítjük az utófeltételt, elhagyjuk belőle $\beta(i)$-t:

P : ($Q \wedge i \geq m \wedge \forall j \in [m..i - 1] : \neg\beta(j)$)

1) $Q' = (Q \wedge i = m)$,

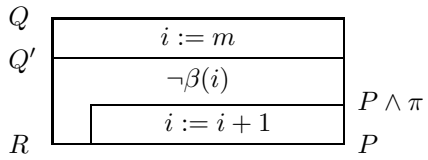
2) $\pi = \neg\beta(i)$,

3) Legyen $N \geq m$ tetszőlegesen rögzített olyan szám, amelyre $\beta(N)$ igaz(ilyen az előfeltétel miatt létezik). Ekkor $t = N - i$.

5) az $i := i + 1$ értékadás csökkenti a terminálófüggvény értékét,

4) $P \wedge \pi \Rightarrow lf(i := i + 1, P)$.

Tétel: Az alábbi struktogram formában megadott program megoldása a fent specifikált feladatnak:



Bizonyítás: A tétel a levezetési szabályokból, és a specifikáció tételéből a fentiek szerint következik. □

A fenti program csak akkor megoldása a feladatnak, ha a β tulajdonság megengedett feltétel. Ha ez nem így van, akkor másik megoldást kell keresnünk. Válasszuk az alábbi invariánst (a feladat marad ugyanaz!):

$P : (Q \wedge i \geq m - 1 \wedge (\forall j \in [m, i - 1] : \neg\beta(j)) \wedge l = \exists j \in [m, i] : \beta(j))$

Ekkor:

1) $Q' = (Q \wedge i = m - 1 \wedge l = hamis)$,

2) $\pi = \neg l$,

3) Legyen $N \geq m$ tetszőlegesen rögzített olyan szám, amelyre $\beta(N)$ igaz(ilyen az előfeltétel miatt létezik). Ekkor $t = N - i$.

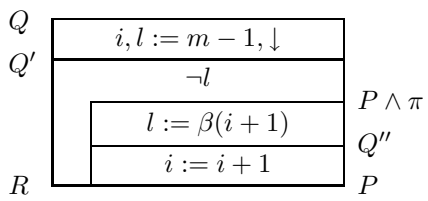
5) az $i := i + 1$ értékadás csökkenti a terminálófüggvény értékét,

4) A ciklusmag szekvencia lesz, melynek közbülső feltétele:

   $Q'' : (Q \wedge i + 1 \geq m - 1 \wedge (\forall j \in [m, i] : \neg\beta(j)) \wedge l = \exists j \in [m, i + 1] : \beta(j))$

   és így a ciklusmag első fele az $l := \beta(i + 1)$ értékadás lesz.

Tétel: Ekkor az alábbi program is megoldása a specifikált feladatnak.



Tegyük fel, hogy $\beta = \gamma \vee \delta$ és létezik $i \geq m$, hogy $\beta(i)$. Keressük meg a legelső $i \geq m : \gamma(i)$ elemet (ha van olyan), amely előtt (m-től kezdve) nem volt igaz δ! Ennek a változatnak már a specifikációja is más lesz, hiszen a feladat is megváltozott.

$$A = \underset{m}{\mathbb{Z}} \times \underset{i}{\mathbb{Z}} \times \underset{u}{\mathbb{L}}$$
$$B = \underset{m'}{\mathbb{Z}}$$
Q : $(m = m' \wedge \exists j \geq m : \beta(j))$
R : $(Q \wedge u = (\exists j \geq m : \gamma(j) \wedge \forall k \in [m..j-1] : \neg\delta(k)) \wedge$
$u \rightarrow (i \geq m \wedge \gamma(i) \wedge \forall j \in [m..i-1] : \neg\beta(j)))$

A feladatot megoldó ciklus invariánsa:

P : $(Q \wedge i \geq m-1 \wedge u = (\exists j \in [m..i] : \gamma(j)) \wedge v = (\exists j \in [m..i] : \delta(j)) \wedge$
$\forall j \in [m, i-1] : \neg\beta(j))$

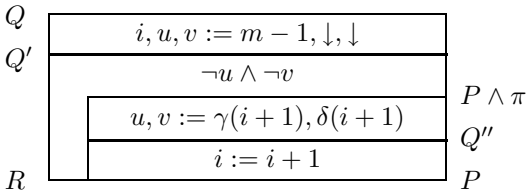
Ekkor:

1) $Q' = (Q \wedge i = m-1 \wedge u = hamis \wedge v = hamis)$,
2) $\pi = \neg u \wedge \neg v$,
3) Legyen $N \geq m$ tetszőlegesen rögzített olyan szám, amelyre $\beta(N)$ igaz (ilyen az előfeltétel miatt létezik). Ekkor $t = N - i$.
5) az $i := i + 1$ értékadás csökkenti a terminálófüggvény értékét,
4) A ciklusmag szekvencia lesz, melynek közbülső feltétele ($lf(i := i+1, P)$):

   Q'' : $(Q \wedge i+1 \geq m-1 \wedge u = (\exists j \in [m..i+1] : \gamma(j)) \wedge$
   $v = (\exists j \in [m..i+1] : \delta(j)) \wedge \forall j \in [m, i] : \neg\beta(j))$

   és így a ciklusmag első fele az $u, v := \gamma(i+1), \delta(i+1)$ értékadás lesz.

Tétel: Az alábbi struktogram formában megadott program megoldása a fent specifikált feladatnak:



A fenti feladatnak van egy speciális esete, amikor δ tulajdonság azt mondja ki, hogy még nem értünk el egy n számot, azaz egy intervallumon kell keresni. Erre a speciális esetre adunk egy másik megoldást is.

$$A = \underset{m}{\mathbb{Z}} \times \underset{n}{\mathbb{Z}} \times \underset{i}{\mathbb{Z}} \times \underset{l}{\mathbb{L}}$$
$$B = \underset{m'}{\mathbb{Z}} \times \underset{n'}{\mathbb{Z}}$$
Q : $(m = m' \wedge n = n' \wedge m \leq n+1)$
R : $(Q \wedge l = (\exists j \in [m..n] : \beta(j)) \wedge l \rightarrow (i \in [m..n] \wedge \beta(i) \wedge$
$\forall j \in [m..i-1] : \neg\beta(j)))$

Legyen a ciklus invariáns tulajdonsága:

P : $(Q \wedge i \in [m-1..n] \wedge l = (\exists j \in [m..i] : \beta(j)) \wedge \forall j \in [m..i-1] : \neg\beta(j))$

Ekkor:

1) $Q' = (Q \wedge i = m - 1 \wedge l = hamis)$,

2) $\pi = \neg l \wedge i \neq n$,

3) $t = n - i$,

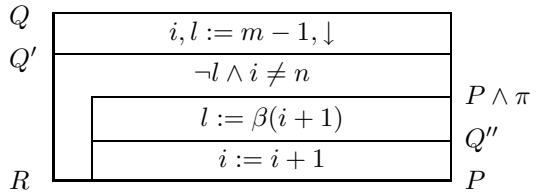
5) az $i := i + 1$ értékadás csökkenti a terminálófüggvény értékét,

4) A ciklusmag szekvencia lesz, melynek közbülső feltétele ($lf(i := i + 1, P)$):

   $Q'' : (Q \wedge i + 1 \in [m - 1..n] \wedge l = (\exists j \in [m..i + 1] : \beta(j)) \wedge \forall j \in [m..i] : \neg\beta(j))$

   és így a ciklusmag első fele az $l := \beta(i + 1)$ értékadás lesz.

Tétel: Az alábbi struktogram formában megadott program megoldása a fent specifikált feladatnak:



9.6. Logaritmikus keresés

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

$A = \underset{m}{\mathbb{Z}} \times \underset{n}{\mathbb{Z}} \times \underset{h}{\mathcal{H}} \times \underset{i}{\mathbb{Z}} \times \underset{l}{\mathbb{L}}$

$B = \underset{m'}{\mathbb{Z}} \times \underset{n'}{\mathbb{Z}} \times \underset{h'}{\mathcal{H}}$

$Q : (m = m' \wedge n = n' \wedge h = h' \wedge m \leq n + 1 \wedge \forall k, j \in [m..n] :$
$(k < j) \to (f(k) \leq f(j)))$

$R : (Q \wedge l = (\exists j \in [m..n] : f(j) = h) \wedge l \to (i \in [m..n] \wedge f(i) = h))$

A monotonitást felhasználva az intervallumot mindkét végéről szűkítjük az invariánsban:

$P : (Q \wedge [u..v] \subseteq [m..n] \wedge \forall j \in [m..n] \setminus [u..v] : f(j) \neq h \wedge$
$l \to (i \in [u..v] \wedge f(i) = h))$

Ekkor

1) $Q' = (Q \wedge u = m \wedge v = n \wedge l = hamis)$,

2) $\pi = \neg l \wedge u \leq v$,

3) Informálisan fogalmazva jelölje t a még megvizsgálandó elemek számát. Ezt egy esetszétválasztással adhatjuk meg:

$$t = \begin{cases} v - u + 1, & \text{ha } \neg l \\ 0, & \text{ha } l \end{cases}$$

4) A ciklusmag legyen egy szekvencia, melynek közbülső feltétele:

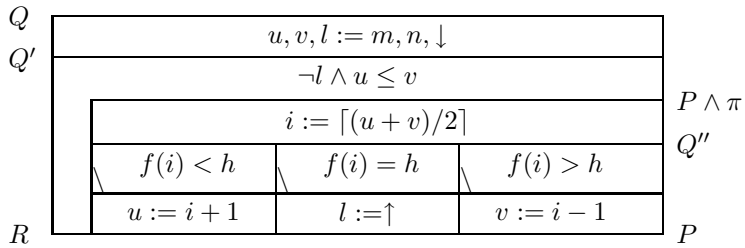
$$Q'' : (Q \wedge [u..v] \subseteq [m..n] \wedge \forall j \in [m..n] \setminus [u..v] : f(j) \neq h \wedge \neg l \wedge (i \in [u..v]))$$

Ekkor a szekvencia első fele lehetne az $i :\in [u..v]$ értékkiválasztás. Hatékonysági szempotokat is fi gyelembe véve azonban válasszuk az intervallum középső elemét: $i := \lceil (u+v)/2 \rceil$. A ciklusmag második felében három eset lehetséges:

- $f(i) < h$: ekkor az $u := i + 1$ értékadás az invariánst megtartja,
- $f(i) = h$: ekkor megtaláltuk a keresett elemet, tehát $l := igaz$,
- $f(i) > h$: ekkor az $v := i - 1$ értékadás az invariánst megtartja.

5) Egyszerűen ellenőrizhető, hogy a fenti elágazás mindhárom ága csökkenti a terminál függvényt.

Tétel: Az alábbi struktogram formában megadott program megoldása a fent specifi kált feladatnak:



Bizonyítás: A tétel a fenti levezetés következménye.

10. fejezet

Függvényérték kiszámítása

A továbbiakban bizonyos speciális függvények helyettesítési értékének kiszámításával fogunk foglalkozni. Tegyük fel, hogy van egy $f : X \to Y$ függvényünk, ahol X és Y tetszőleges halmazok. A feladat specifikációja tehát:

$$\begin{array}{lll} A = & X & \times\ Y \\ & x & \quad y \end{array}$$

$$\begin{array}{ll} B = & X \\ & x' \end{array}$$

$Q : (x = x')$

$R : (y = f(x'))$

Természetesen ha semmi mást nem tudunk a függvényről, akkor a fenti specifikációhoz nem tudunk igazi megoldóprogramot adni. Ezért az elkövetkezőkben további feltételezésekkel fogunk élni.

10.1. Függvénykompozícióval adott függvény kiszámítása

Tegyük fel, hogy $f = h \circ g$, ahol $g : X \to Z$ és $h : Z \to Y$ függvények.

Tétel: Ekkor a feladat megoldható az alábbi szekvenciával:

$z := g(x)$
$y := h(z)$

Bizonyítás: Kibővítjük az állapotteret egy újabb (Z típusú) komponenssel, melynek változója legyen z. A szekvencia közbülső feltétele legyen

$Q' : (z = g(x'))$.

Ekkor a szekvencia levezetési szabálya alapján a megoldás triviálisan teljesül.

□

10.2. Esetszétválasztással adott függvény kiszámítása

Legyenek $\pi_1, \pi_2, \ldots, \pi_n : X \to \mathbb{L}$ feltételek és $g_1, g_2, \ldots, g_n : X \to Y$ függvények, és tegyük fel, hogy a π_i feltételek lefedik az X halmazt. Legyen $f : X \to Y$ az alábbi módon definiálva:

$$f(x) = \begin{cases} g_1(x), & \text{ha } \pi_1(x) \\ g_2(x), & \text{ha } \pi_2(x) \\ \vdots & \vdots \\ g_n(x), & \text{ha } \pi_n(x) \end{cases}$$

Tétel: Ekkor az f függvény értéke kiszámolható az alábbi elágazással:

$\pi_1(x)$	$\pi_2(x)$	$\ldots$	$\pi_n(x)$
$y := g_1(x)$	$y := g_2(x)$	$\ldots$	$y := g_n(x)$

Bizonyítás: Az elágazás levezetési szabálya alapján a megoldás triviálisan teljesül. □

10.3. Rekurzív formulával adott függvény kiszámítása

Legyen H egy tetszőleges halmaz, $k > 0$ egy egész szám, továbbá $F : \mathbb{Z} \times H^k \to H$ függvény, $t_0, t_{-1}, \ldots, t_{-k+1} \in H$ rögzített, és definiáljuk az $f : \mathbb{Z} \to H$ függvényt az alábbi módon:

$$\begin{aligned} f(0) &= t_0, \\ f(-1) &= t_{-1}, \\ &\vdots \quad \vdots \\ f(-k+1) &= t_{-k+1} \end{aligned}$$

továbbá $\forall i \geq 0$:

$$f(i+1) = F(i+1, f(i), ..., f(i-k+1))$$

Feladatunk az, hogy meghatározzuk az f függvény $n \geq 0$ helyen felvett értékét.

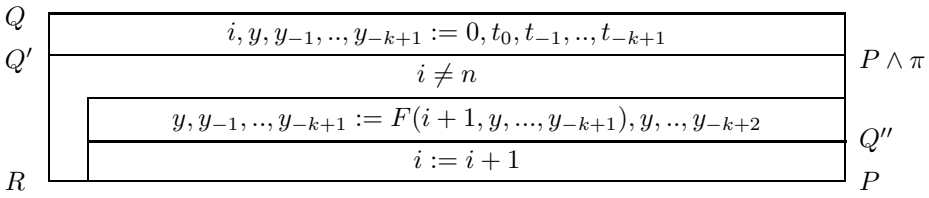
$$\begin{array}{lccc} A = & \mathbb{Z} & \times & H \\ & n & & y \end{array}$$

$$\begin{array}{ll} B = & \mathbb{Z} \\ & n' \end{array}$$

$Q : (n = n' \wedge n \geq 0)$

$R : (Q \wedge y = f(n))$

Tétel: Az alábbi struktogrammal adott program megoldása a specifikált feladatnak:



Bizonyítás:

A tétel bizonyításához elegendő, ha a fenti programot levezetjük, hiszen ekkor az állítás a specifikáció tételéből és a levezetési szabályokból következik. Legyen tehát a megoldóprogram egy ciklus, melynek invariánsa:

$P : (Q \wedge i \in [0..n] \wedge y = f(i), y_{-1} = f(i-1), \ldots, y_{-k+1} = f(i-k+1))$

Vizsgáljuk meg a ciklus levezetési szabályának feltételrendszerét:

1) Mivel az eredeti Q feltételre $Q \Rightarrow P$ nem áll fenn, a ciklus előfeltétele az alábbi

   $Q' : (Q \wedge i = 0 \wedge y = t_0, y_{-1} = t_{-1}, \ldots, y_{-k+1} = t_{-k+1})$

   állítás lesz. Egyszerűen látható, hogy a fenti program elején található szimultán értékadás Q-ból Q'-be jut.

2) $\pi = (i \neq n)$,

3) $t = n - i$,

5) a $i := i + 1$ értékadás csökkenti a terminálófüggvény értékét,

4) Ha felírjuk az $i := i + 1$ értékadás P-re vonatkozó leggyengébb előfeltételét:

   $Q'' : (Q \wedge i + 1 \in [0..n] \wedge y = f(i+1), y_{-1} = f(i), \ldots, y_{-k+1} = f(i-k+2))$

   akkor az értékadás leggyengébb előfeltételére vonatkozó szabály alapján egyszerű behelyettesítéssel verifikálható, hogy a ciklusmag első fele $P \wedge \pi$-ből Q''-be jut.

□

Megjegyezzük még, hogy a 0 kezdőpont választása önkényes, bármilyen tetszőleges egész számtól kezdve definiálható egy függvény, és akkor értelemszerűen a program ciklusváltozója is arról az értékről indítandó.

10.4. Elemenként feldolgozható függvény

A továbbiakban legyenek H_1 és H_2 tetszőleges halmazok, X és Y pedig az alábbi formában felírható halmazok:

$$\begin{aligned} X &= X_1 \times ... \times X_n \\ Y &= Y_1 \times ... \times Y_m \end{aligned}$$

ahol $X_i = \{x \in 2^{H_1} : |x| < \infty\} \quad i \in [1..n]$ és $Y_i = \{y \in 2^{H_2} : |y| < \infty\} \quad i \in [1..m]$. Amint az a fenti leírásból kiderül az X az összes olyan halmaz n-est tartalmazza, amelyeknek minden komponense az adott H_1 halmaz véges részhalmaza. Hasonlóan az Y elemei pedig az olyan halmaz m-esek, amelyek H_2-beli véges részhalmazokból állnak.

Definíció: TELJESEN DISZJUNKT FELBONTÁS

Azt mondjuk, hogy $\overline{x}, \overline{\overline{x}} \in X$ *teljesen diszjunkt felbontása* $x \in X$ -nek, ha

i) $\forall i \in [1..n] : x_i = \overline{x}_i \cup \overline{\overline{x}}_i$ és

ii) $\forall i, j \in [1..n] : \overline{x}_i \cap \overline{\overline{x}}_j = \emptyset$.

Vegyük észre, hogy ha X egydimenziós, akkor a teljesen diszjunkt felbontás megegyezik a diszjunkt felbontással, de többdimenziós esetben a teljesen diszjunkt felbontás egy jóval erősebb feltételt jelent.

Definíció: ELEMENKÉNT FELDOLGOZHATÓ FÜGGVÉNY

Legyen $f : X \to Y$. Ha minden $x \in X$ minden $\overline{x}, \overline{\overline{x}}$ teljesen diszjunkt felbontására

i) $\forall i \in [1..m] : f_i(\overline{x}) \cup f_i(\overline{\overline{x}}) = f_i(x)$ és

ii) $\forall i \in [1..m] : f_i(\overline{x}) \cap f_i(\overline{\overline{x}}) = \emptyset$,

akkor f -et *elemenként feldolgozhatónak* nevezzük.

Példa: Legyen H egy tetszőleges halmaz, $X_1 = X_2 = Y = \{x \in 2^H : |x| < \infty\}$, $f : X_1 \times X_2 \to Y$, $f((x_1, x_2)) = x_1 \cup x_2$. Ekkor f elemenként feldolgozható, ui: tekintsük az (x_1, x_2) halmazpár egy tetszőleges $\overline{(x_1, x_2)}$, $\overline{\overline{(x_1, x_2)}}$ teljesen diszjunkt felbontását. Ekkor a teljesen diszjunkt felbontás definíciója alapján:

$$\begin{array}{llll} \overline{x}_1 \cup \overline{\overline{x}}_1 = x_1, & \overline{x}_2 \cup \overline{\overline{x}}_2 = x_2 \\ \overline{x}_1 \cap \overline{\overline{x}}_1 = \emptyset, & \overline{x}_2 \cap \overline{\overline{x}}_2 = \emptyset \\ \overline{x}_1 \cap \overline{\overline{x}}_2 = \emptyset, & \overline{x}_2 \cap \overline{\overline{x}}_1 = \emptyset \end{array}$$

Vizsgáljuk most meg az elemenként feldolgozhatóság két kritériumát:

1. $f(\overline{(x_1, x_2)}) \cup f(\overline{\overline{(x_1, x_2)}}) = (\overline{x}_1 \cup \overline{x}_2) \cup (\overline{\overline{x}}_1 \cup \overline{\overline{x}}_2) = (\overline{x}_1 \cup \overline{\overline{x}}_1) \cup (\overline{x}_2 \cup \overline{\overline{x}}_2) = x_1 \cup x_2 = f((x_1, x_2))$,

2. $f(\overline{(x_1, x_2)}) \cap f(\overline{\overline{(x_1, x_2)}}) = (\overline{x}_1 \cup \overline{x}_2) \cap (\overline{\overline{x}}_1 \cup \overline{\overline{x}}_2) = (\overline{x}_1 \cap \overline{\overline{x}}_1) \cup (\overline{x}_1 \cap \overline{\overline{x}}_2) \cup (\overline{x}_2 \cap \overline{\overline{x}}_1) \cup (\overline{x}_2 \cap \overline{\overline{x}}_2) = \emptyset$.

Tehát a – kétváltozós – unió elemenként feldolgozható halmazfüggvény.

A továbbiakban tehát egy elemenként feldolgozható függvény helyettesítési értékének kiszámításával fogunk foglalkozni.

Mielőtt belekezdenénk a feladat specifikálásába és megoldásába bevezetünk két olyan halmazokra vonatkozó parciális értékadást, amelyeket aztán a megoldó programokban primitív műveletnek tekintünk.

- Legyen H egy tetszőleges halmaz, és definiáljuk az $f_{\widetilde{\cup}} : 2^H \times H \to 2^H$ parciális függvényt:
$$f_{\widetilde{\cup}}(h, e) = H \cup \{e\}, \text{ha } e \notin H.$$
A fenti függvényt kiszámító $h := f_{\widetilde{\cup}}(h, e)$ parciális értékadást a továbbiakban $h := h \mathbin{\widetilde{\cup}} e$-vel fogjuk jelölni.

- Hasonlóan legyen H egy tetszőleges halmaz, és definiáljuk az $f_{\widetilde{\simeq}} : 2^H \times H \to 2^H$ parciális függvényt:
$$f_{\widetilde{\simeq}}(h, e) = H \setminus \{e\}, \text{ha } e \in H.$$
A fenti függvényt kiszámító $h := f_{\widetilde{\simeq}}(h, e)$ parciális értékadást a továbbiakban $h := h \mathbin{\widetilde{\simeq}} e$-vel fogjuk jelölni.

10.4.1. Egyváltozós-egyértékű eset

Először vizsgáljuk meg azt az esetet, amikor mind X, mind Y egykomponensű, azaz $m = n = 1$. Ekkor az f függvény egy halmazhoz egy másik halmazt rendel.

$$\begin{array}{lll} A = & X & \times\ Y \\ & x & \quad y \end{array}$$

$$\begin{array}{ll} B = & X \\ & x' \end{array}$$

$Q : (x = x')$

$R : (y = f(x'))$

Oldjuk meg a feladatot ciklussal: az invariánsban azt fogalmazzuk meg, hogy az x halmaz a még feldolgozandó elemeket, az y halmaz pedig a már feldolgozott elemek f szerinti képeinek unióját tartalmazza, azaz

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

Vizsgáljuk meg a ciklus levezetési szabályának feltételeit:

1) Q-ból az $y = \emptyset$ fennállása esetén következik P, ezért a ciklus elé az $y := \emptyset$ értékadás kerül.

2) Az invariánsból $f(x) = \emptyset$ esetén következik az utófeltétel, ám ez jó eséllyel nem egy megengedett feltétel (hiszen éppen f-et akarjuk kiszámítani). Vegyük észre azonban, hogy f elemenkénti feldolgozhatósága miatt $x = \emptyset$ esetén $f(x) = \emptyset$ is teljesűl (az üres halmaznak két üres halmaz egy teljesen diszjunkt felbontása). Tehát a ciklusfeltétel: $\pi = x \neq \emptyset$.

3) Ha (a ciklusfeltétel szerint) x nem üres, akkor terminálló függvénynek választható x számossága, azaz $t = |x|$.

5) x számosságát úgy tudjuk csökkenteni, ha elhagyunk belőle egy – benne levő – elemet. Ezt megtehetjük az imént bevezetett parciális értékadással: $x := x \simeq e$.

4) Írjuk fel a fenti parciális értékadás P-re vonatkozó leggyengébb előfeltételét:

   $Q'' : (y \cup f(x \setminus \{e\}) = f(x') \wedge y \cap f(x \setminus \{e\}) = \emptyset \wedge e \in x)$

   Jól látható, hogy ez $P \wedge \pi$-ből nem következik. Vegyük azonban észre, hogy ha e egy x-beli elem, akkor $\{e\}$ és $x \setminus \{e\}$ x-nek egy teljesen diszjunkt felbontása, tehát f elemenkénti feldolgozhatósága miatt:

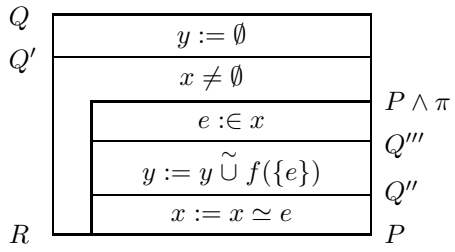
$$\begin{array}{rcl} f(\{e\}) \cup f(x \setminus \{e\}) & = & f(x) \\ f(\{e\}) \cap f(x \setminus \{e\}) & = & \emptyset \end{array}$$

   Így az $y := y \mathbin{\widetilde{\cup}} f(\{e\})$ értékadás már majdnem elegendő, hiszen

   $lf(y := y \mathbin{\widetilde{\cup}} f(\{e\}), Q'') = (y \cup f(\{e\}) \cup f(x \setminus \{e\}) = f(x') \wedge (y \cup f(\{e\})) \cap f(x \setminus \{e\}) = \emptyset \wedge e \in x)$.

   Ezt a feltételt összevetve $P \wedge \pi$-vel látható, hogy már csak az $e \in x$ állítást kell teljesítenünk. Ezt viszont megtehetjük az $e :\in x$ értékkiválasztással, amelynek a fenti állításra vonatkozó leggyengébb előfeltétele $P \wedge \pi$.

Tétel: Ekkor az alábbi program megoldása a specifikált feladatnak:



Bizonyítás: A tétel a fenti levezetésből következik.

10.4.2. Kétváltozós-egyértékű eset

Legyen $f : X \times Y \to Z \ \ (X, Y, Z \subseteq 2^H)$ elemenként feldolgozható függvény.

$$\begin{array}{lccccc} A = & X & \times & Y & \times & Z \\ & x & & y & & z \end{array}$$

$$\begin{array}{lccc} B = & X & \times & Y \\ & x' & & y' \end{array}$$

$Q : (x = x' \wedge y = y')$

$R : (z = f(x', y'))$

Tétel: Ekkor az alábbi program megoldása a specifikált feladatnak:

$z := \emptyset$		
$x \neq \emptyset \vee y \neq \emptyset$		
$e :\in (x \cup y)$		
$e \in x \wedge e \notin y$	$e \in x \wedge e \in y$	$e \notin x \wedge e \in y$
$z := z \ \widetilde{\cup} \ f(\{e\}, \emptyset)$	$z := z \ \widetilde{\cup} \ f(\{e\}, \{e\})$	$z := z \ \widetilde{\cup} \ f(\emptyset, \{e\})$
$x := x \simeq e$	$x := x \simeq e$	$y := y \simeq e$
	$y := y \simeq e$	

Bizonyítás: A tétel az egyváltozós esettel analóg módon levezethető, ha invariáns tulajdonságnak az alábbi állítást:

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

terminálófüggvénynek pedig $t = |x \cup y|$-t választjuk. □

10.4.3. Egyváltozós kétértékű eset

Legyen $f : X \to Y \times Z \ \ (X, Y, Z \subseteq 2^H, f_1 : X \to Y, f_2 : X \to Z, f = (f_1, f_2))$ elemenként feldolgozható függvény.

$$\begin{array}{lccccc} A = & X & \times & Y & \times & Z \\ & x & & y & & z \end{array}$$

$$\begin{array}{lc} B = & X \\ & x' \end{array}$$

$Q : (x = x')$

$R : (y = f_1(x') \wedge z = f_2(x'))$

Tétel: Ekkor az alábbi program megoldása a specifikált feladatnak:

$y, z := \emptyset, \emptyset$	
$x \neq \emptyset$	
	$e :\in x$
	$y, z := y \mathbin{\widetilde{\cup}} f_1(\{e\}), z \mathbin{\widetilde{\cup}} f_2(\{e\})$
	$x := x \simeq \{e\}$

Bizonyítás: A tétel levezetése az egyértékű esettől csak az invariáns tulajdonság megválasztásában tér el:

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

A terminálló függvény marad, és a levezetés lépései is megegyeznek.□

10.4.4. Általános változat

Legyenek n, m rögzített természetes számok, $f : X_1 \times \cdots \times X_n \to Y_1 \times \cdots \times Y_m (X_i, Y_j \in 2^H,\ (i \in [1..n], j \in [1..m]))$ elemenként feldolgozható függvény, és legyenek az $f_j : X_1 \times \cdots \times X_n \to Y_j\ \ (j \in [1..m])$ függvények az f komponensfüggvényei, azaz $f = (f_1, \ldots, f_m)$.

$$\begin{array}{lccccccccc} A = & X_1 & \times & \ldots & \times & X_n & \times & Y_1 & \times \ldots \times & Y_m \\ & x_1 & & & & x_n & & y_1 & & y_n \end{array}$$

$$\begin{array}{lccccc} B = & X_1 & \times & \ldots & \times & X_n \\ & x'_1 & & & & x'_n \end{array}$$

$Q : (x_1 = x'_1 \wedge \cdots \wedge x_n = x'_n)$

$R : (y_1 = f_1(x'_1, \ldots, x'_n) \wedge \cdots \wedge y_m = f_m(x'_1, \ldots, x'_n))$

$y_1, ..., y_m := \emptyset, ..., \emptyset$		
$\bigcup_{i=1}^{n} x_i \neq \emptyset$		
	$e :\in \bigcup_{i=1}^{n} x_i$	
$\ldots$	$e \in x_{i_1} \wedge \ldots e \in x_{i_k} \wedge e \notin x_{i_{k+1}} \wedge e \notin x_{i_n}$	$\ldots$
$\ldots$	$y_1, \ldots, y_m :=$ $y_1 \mathbin{\widetilde{\cup}} f_1(sl(\{i_1 \ldots i_k\}, e)), \ldots, y_m \mathbin{\widetilde{\cup}} f_m(sl(\{i_1 \ldots i_k\}, e))$ $x_{i_1}, \ldots, x_{i_k} := x_{i_1} \simeq e, \ldots, x_{i_k} \simeq e$	$\ldots$

ahol $i_1, \ldots, i_n$ az $1, \ldots, n$ permutációja, $sl : 2^{\{1,\ldots,n\}} \times H \to X_1 \times \cdots \times X_n$,

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

Az elágazás "ágainak" száma $2^n - 1$.

Bizonyítás: A tétel az alábbi invariáns tulajdonsággal és terminálló függvénnyel formálisan levezethető (a levezetést annak bonyolultsága miatt elhagyjuk):

$$P : (\forall j \in [1..m] : (y_j \cup f_j(x_1, \ldots, x_n) = f_j(x'_1, \ldots, x'_n) \wedge$$
$$y_j \cap f_j(x_1, \ldots, x_n) = \emptyset \wedge \forall i, k \in [1..n] : (x'_i \setminus x_i) \cap x_k = \emptyset))$$

$$t = \left| \bigcup_{i=1}^{n} x_i \right|$$

□

11. fejezet

Visszalépéses keresés

Legyen $n \in \mathbb{N}$, és $n > 1$. Legyenek U_i $(i \in [1..n])$ tetszőleges véges, nem üres halmazok ($0 < \sigma_i = |U_i| < \infty$). $U = U_1 \times \cdots \times U_n$.

Legyen $\varrho : U \to \mathbb{L}$, amely felbontható $\varrho_i : U \to \mathbb{L}$ $(i \in [0..n])$ tulajdonságok sorozatára az alábbi módon:

1. $\varrho_0 = \uparrow$;
2. $\forall i \in [0..n-1] : \forall u \in U : \varrho_{i+1}(u) \to \varrho_i(u)$;
3. $\forall i \in [1..n] : \forall u, v \in U : (\forall j \in [1..i] : u_j = v_j) \to \varrho_i(u) = \varrho_i(v)$;
4. $\varrho = \varrho_n$.

A feladat annak eldöntése, hogy létezik-e olyan elem U-ban, amelyre teljesül a ϱ feltétel, és ha igen, adjunk meg egy ilyen elemet.

$$
\begin{array}{l}
A = U \times \mathbb{L} \\
\quad\;\; u \quad\; l \\
B = \{\mathcal{X}\} \\
Q : \uparrow \\
R : (l = \exists v \in U : \varrho(v) \wedge l \to (u \in U \wedge \varrho(u)))
\end{array}
$$

Számozzuk meg U elemeit az alábbi módon: Minden U_i halmaz elemeit számozzuk meg nullától $\sigma_i - 1$-ig. Ezután U minden u eleméhez van egy $(i_1, \ldots, i_n)$ rendezett n-es, amelyre $u = (u_{i_1}, \ldots, u_{i_n})$, ahol $0 \leq i_1 < \sigma_1, \ldots, 0 \leq i_n < \sigma_n$. Ezen megszámozás egy lexikografi kus rendezést defi niál U-n.

Legyen $\mathcal{N} = [0..\sigma_1 - 1] \times \cdots \times [0..\sigma_n - 1]$. Ekkor a fenti megszámozás egy bijekciót létesít $\mathcal{N}$ és U között. Jelölje ezt az $\mathcal{N} \to U$ leképezést φ.

Vegyük észre, hogy az $\mathcal{N}$ halmaz elemei felfoghatók vegyesalapú számrendszerben felírt számként is. Ez alapján egy $\nu \in \mathcal{N}$ n-es számértéke:

$$
\begin{aligned}
f(\nu) &= \sum_{i=1}^{n} \nu_i * Q_i, \qquad \text{ahol:} \\
Q_i &= \prod_{j=i+1}^{n} \sigma_j \quad (i \in [1..n])
\end{aligned}
$$

A bevezetett jelölésekkel a feladatot újra specifikáljuk, és most már megkövetelhetjük azt is, hogy ha létezik keresett tulajdonságú elem, akkor az első ilyet adjuk meg:

$$\begin{array}{l} A = \underset{\nu}{\mathcal{N}} \times \underset{l}{\mathbb{L}} \\ B = \{\mathcal{X}\} \\ Q: \uparrow \\ R: (l = (\exists \mu \in \mathcal{N} : \varrho(\varphi(\mu))) \wedge \\ \quad l \rightarrow (\varrho(\varphi(\nu)) \wedge \forall \mu \in \mathcal{N} : f(\mu) < f(\nu) \rightarrow \neg\varrho(\varphi(\mu)))) \end{array}$$

Ha nem használjuk ki a ϱ speciális tulajdonságait, akkor a fenti feldat megoldható lineáris kereséssel, a $[0..|\mathcal{N}| - 1]$ intervallumon. Vizsgáljuk meg, hogy hogyan használhatnánk ki ϱ specialitását!

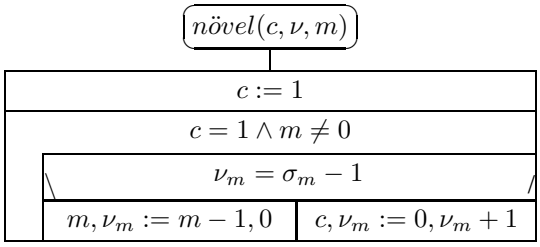
Ha $\varrho_i(\varphi(\nu))$ igaz és $\varrho_{i+1}(\varphi(\nu))$ pedig hamis, akkor minden olyan $\nu' \in \mathcal{N}$-re, amelynek első $i+1$ komponense megegyezik ν első $i+1$ komponensével, $\varrho_{i+1}(\varphi(\nu'))$ is hamis lesz. Ezért ha az $i+1$-edik pozíció után a ν csak nullákat tartalmaz, akkor a keresésben nagyobbat léphetünk, és növelhetjük ν-t az $i+1$-edik pozíción.

Az algoritmus még egyszerűbbé tehető, ha a ν-t kiegészítjük egy túlcsordulás bittel, amelynek 1 értéke azt jelzi, hogy ν értéke már nem növelhető.

Terjesszük ki az f függvényt az alábbi módon:
$f : \{0,1\} \times \mathcal{N} \rightarrow \mathbb{N}_0$,

$$\begin{aligned} f(c,\nu) &= c * Q_0 + \sum_{i=1}^{n} \nu_i * Q_i, \qquad \text{ahol:} \\ Q_0 &= \prod_{j=1}^{n} \sigma_j \end{aligned}$$

$$\begin{array}{l} A_{növel} = \underset{\nu}{\mathcal{N}} \times \underset{m}{\mathbb{N}_0} \times \underset{c}{\{0,1\}} \\ B_{növel} = \underset{\nu'}{\mathcal{N}} \times \underset{m'}{\mathbb{N}_0} \\ Q_{növel} : (\nu = \nu' \wedge m = m' \wedge m' \in [1..n] \wedge \forall i \in [m'+1..n] : \nu_i = 0i) \\ R_{növel} : (f(c,\nu) = f(\nu') + Q_{m'} \wedge m \in [0..m'] \wedge \\ \qquad \forall i \in [m+1..n] : \nu_i = 0) \end{array}$$

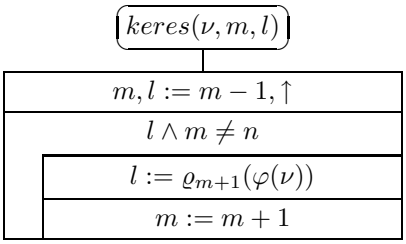


$$P_{növel} : (f(\nu) + c * Q_m = f(\nu') + Q_{m'} \wedge m \in [0..m'] \wedge \forall i \in [m+1..n] : \nu_i = 0 \wedge \forall i \in [1..m-1] : \nu_i = \nu'_i)$$

A fenti meggondolások másik következménye, hogy célszerű egy olyan művelet bevezetése is amely amellett, hogy $\varrho(\varphi(\nu))$-t eldönti, azt a legkisebb indexet is megadja, amelyre $\varrho_i(\varphi(\nu))$ hamis.

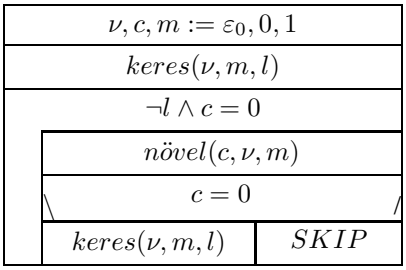
$$A_{keres} = \underset{\nu}{\mathcal{N}} \times \underset{m}{\mathbb{N}_0} \times \underset{l}{\mathbb{L}}$$
$$B_{keres} = \underset{\nu'}{\mathcal{N}} \times \underset{m}{\mathbb{N}_0}$$
$$Q_{keres} : (\nu = \nu' \wedge m = m' \wedge m' \in [1..n] \wedge \varrho_{m'-1}(\varphi(\nu)))$$
$$R_{keres} : (\nu = \nu' \wedge l = \varrho(\varphi(\nu)) \wedge$$
$$\neg l \rightarrow (m \in [m'..n] \wedge \varrho_{m-1}(\varphi(\nu)) \wedge \neg\varrho_m(\varphi(\nu))))$$

Ez a feladat visszavezethető lineáris keresés 2.8-ra.

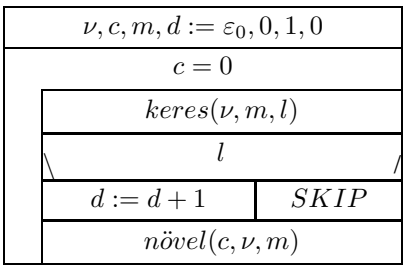


A program levezetéséhez a már bemutatott specifikációt használjuk. Legyen az invariáns tulajdonság:

$$P : (\forall \mu \in \mathcal{N} : 0 \leq f(0, \mu) < f(c, \nu) \rightarrow \neg\varrho(\varphi(\mu)) \wedge l = \varrho(\varphi(\nu)) \wedge$$
$$\neg l \rightarrow (c = 1 \vee m \in [1..n] \wedge \neg\varrho_m(\varphi(\nu)) \wedge \varrho_{m-1}(\varphi(\nu)) \wedge$$
$$\forall i \in [m+1..n] : \nu_i = 0))$$



Megjegyezzük, hogy a visszalépéses kereséshez hasonlóan több visszalépéses technikát alkalmazó algoritmus is levezethető, például a visszalépéses számlálás:



12. fejezet

Programtranszformációk

A továbbiakban olyan esetekkel fogunk foglalkozni, amelyekben a feladat állapotterét kell megváltoztatnunk. Vegyük észre: eddig is csináltunk már ilyet: bevezethetünk új változókat, vagy akár el is hagyhatunk komponenseket.

Az összetettebb esetekben az eredeti állapottér bizonyos komponenseit újakkal helyettesítjük.

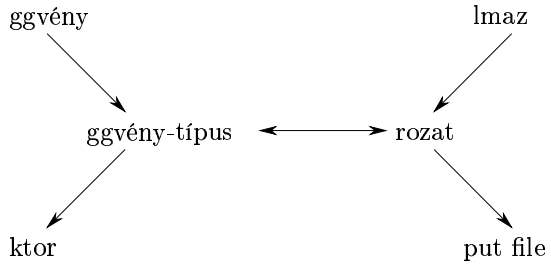
12.1. Koordináta transzformációk

Gyakran előfordul, hogy az állapottér egy komponensének típusát egy kapcsolódó másik típusra kell cserélnünk. Az ilyen transzformációk úgy általánosíthatók, hogy egy leképezést definiálunk a két állapottér egymásnak megfelelő komponensei között.

12.1.1. Típustranszformációk

Típustranszformációról akkor beszélhetünk, ha az állapottér bizonyos komponenseit valami kapcsolódó típusra cseréljük.

A programozási tételek – és általában véve a (feladat, megoldó program) párok – az alábbiakban bemutatásra kerülő transzformációkon keresztül általánosíthatók, ha az állapotterek közötti transzformáció tulajdonképpen típustranszformáció. Ekkor az ábrán látható valamely típuson megoldott program egyszerű szabályok segítségével átvihető egy kapcsolódó típusra.



12.1. ábra. Típusok közötti kapcsolatok

A programozási tételek átírása más típusra. Az alábbi sémák a programozási tételek típusok közötti transzformációjakor elvégzendő helyettesítéseket definiálják. A transzformáció úgy történik, hogy az első típus adott műveletét a második típus megfelelő (azonos sorban levő) műveletére cseréljük.

- **Halmazról** (x, y) **sorozatra** (a, b) (egy input halmaz/sorozat esetén)

  A két típus közötti megfeleltetés: az a és b sorozatok tagjai felsorolják az x és y halmazok elemeit.

$y := \emptyset$	$b := \langle\rangle$
$x \neq \emptyset$	$a.dom \neq 0$
e	$a.lov$
$y \widetilde{\cup}$	$b : hiext^*$
$x \simeq$	$a : lorem$

  A fenti megfeleltetésben szereplő $hiext^*$ művelet a $hiext$ művelet olyan kiterjesztése, amely megengedi, hogy egy legfeljebb egy elemű halmazzal terjesszük ki a sorozatot. Ha a halmaz egy elemű, akkor azzal az elemmel terjeszti ki a sorozatot, míg ha üres, akkor a sorozatot helyben hagyja. Ennek megfelelően az elemenként feldolgozható f függvényről is feltesszük, hogy egy egy elemű halmazhoz legfeljebb egy elemű halmazt rendel hozzá. Az $e :\in x$ művelet helyett az e változónak feleltettük meg az $a.lov$ értéket, mert a lov függvény mindig a sorozat első elemét adja vissza ellentétben az eredeti értékkiválasztással.

  Példa (egyváltozós egyértékű elemenkénti feldolgozás):

$y := \emptyset$	$b := \langle\rangle$
$x \neq \emptyset$	$a.dom \neq 0$
$e :\in x$	$b : hiext(f(\{a.lov\}))$
$y := y \widetilde{\cup} f(\{e\})$	$a : lorem$
$x := x \simeq e$	

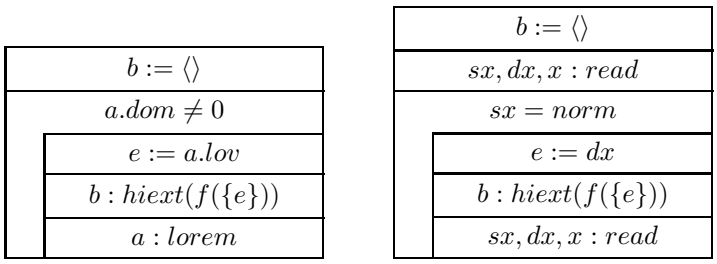
- **Sorozatról** (a) **szekvenciális input file-ra** (lopop,ext) (x) vagy (read) (y): (előreolvasási technika)

  A két típus közötti megfeleltetés: ha az a sorozat nem üres, akkor megegyezik a $loext(x, dx)$ és $loext(y, dy)$ sorozatokkal, míg üres sorozat esetén az x és y sorozatokkal.

	$dx, x : lopop$	$sy, dy, y : read$
$a.lov$	dx	dy
$a : lorem$	$dx, x : lopop$	$sy, dy, y : read$
$a.dom \neq 0$	$dx \neq extr$	$sy = norm$

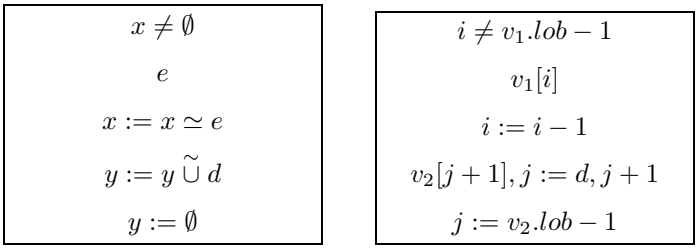
  A transzformált program egy plusz $lopop$ (vagy $read$) művelettel kezdődik.

  Példa (egyváltozós egyértékű elemenkénti feldolgozás):

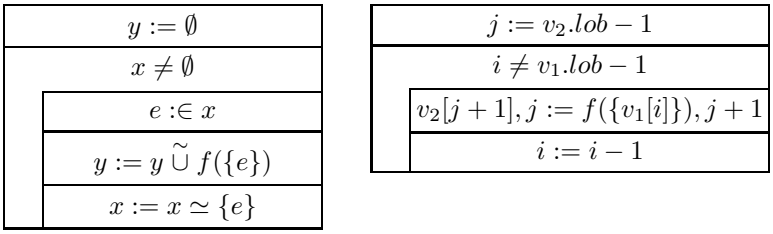


- **Halmazról** (x, y) **vektorra** $(v_1, v_2,\ i \in [v_1.lob, v_1.hib], j \in [v_2.lob, v_2.hib])$

  Feleltessük meg az x (és y) halmaznak a (v_1, i) (ill. (v_2, j)) párt oly módon, hogy az x (ill. y) halmaz elemei a v_1 vektor $v_1.lob..i$ (ill. a v_2 vektor $v_2.lob..j$) indexű elemeivel egyeznek meg.



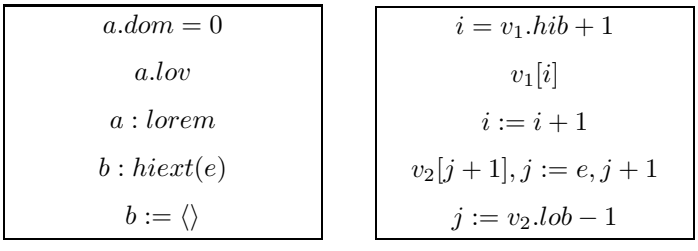
  Példa (egyváltozós egyértékű elemenkénti feldolgozás):



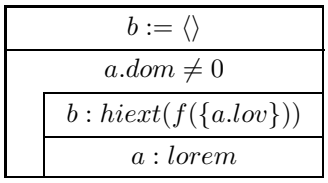
- **Sorozatról** (a, b) **vektorra** $(v_1, v_2,\ i \in [v_1.lob, v_1.hib]\ , j \in [v_2.lob, v_2.hib])$

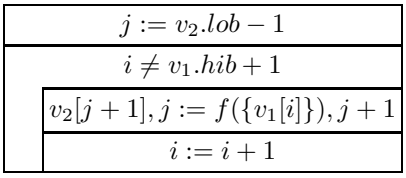
  Sorozatok esetén a sorozat és vektor típusokra megengedett műveletek korlátozzák a két típus közötti megfeleltetés szabad választását: nem használható ugyanaz a leképezés, mint halmazoknál. Válasszunk hát a típusműveletekhez illeszkedő megfeleltetést!

  Feleltessük meg az a sorozatnak a (v_1, i) párt oly módon, hogy az a sorozat tagjai rendre a v_1 vektor $i..v_1.hib$ indexű elemeivel egyeznek meg. A b sorozatnak viszont a (v_2, j) párt úgy feleltessük meg, hogy a b sorozat elemei rendre a v_2 vektor $v_2.lob..j$ indexű elemeivel egyeznek meg.



Példa (egyváltozós egyértékű elemenkénti feldolgozás):

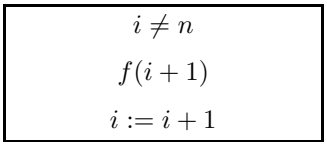


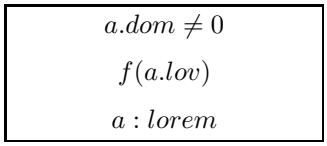


- **Intervallum felett definiált függvényről** ($[m, n]$, $i \in [m, n]$ és a függvény argumentuma $i + 1$) **sorozatra** (a)

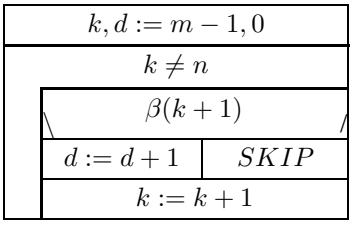
  Első ránézésre úgy tűnhet, hogy kihagytunk egy lépést, a függvénytípust. De vegyük észre, hogy a függvény típusra való áttérés nem igényel transzformációt!

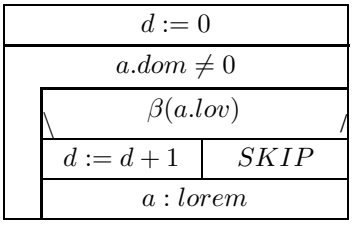
  Itt tulajdonképpen a sorozatot az intervallumnak feleltetjük meg az alábbi módon: tekintsük az a sorozatot egy az $[1..a.dom]$ intervallumon értelmezett függvénynek, ahol a függvényértékek a sorozat megfelelő elemei. Ekkor a tételben szereplő függvénynek tulajdonképpen az $f \circ a$ függvényt tekinthetjük.





Példa (számlálás tétele):





- **Halmazokról** (x, y, z) szigorúan monoton növő **sorozatokra** (a, b, c) a kétváltozós elemenkénti feldolgozás esetén. A kétváltozós elemenkénti feldolgozásnál az okozhat gondot, hogy el kell tudnunk dönteni, hogy egy adott elem melyik sorozatban van benne. Ezért tettük fel, hogy a sorozatok növekvően rendezettek, mert így mindkét sorozat legelső eleme a legkisebb, és ha ezek közül a kisebbiket választjuk, akkor a tartalmazás egyszerűen eldönthető. Egyébként a transzformáció többi része megegyezik az egyváltozós esetnél leírtakkal (itt is szükséges, hogy a függvényérték legfeljebb egyelemű legyen). Természetesen az elem kiválasztása itt is elhagyható, hiszen a lov művelet a sorozatnak mindig ugyanazt az elemét adja vissza.

<table>
<tr><td colspan="3">$c := \langle\rangle$</td></tr>
<tr><td colspan="3">$a.dom \neq 0 \vee b.dom \neq 0$</td></tr>
<tr><td>$(a.dom \neq 0 \wedge b.dom \neq 0 \wedge a.lov < b.lov) \vee b.dom = 0$</td><td>$a.dom \neq 0 \wedge b.dom \neq 0 \wedge a.lov = b.lov$</td><td>$(a.dom \neq 0 \wedge b.dom \neq 0 \wedge a.lov > b.lov) \vee a.dom = 0$</td></tr>
<tr><td>$c : hiext(f(\{a.lov\}, \emptyset))$
$a : lorem$</td><td>$c : hiext(f(\{a.lov\}, \{b.lov\}))$
$a : lorem$
$b : lorem$</td><td>$c : hiext(f(\emptyset, \{b.lov\}))$
$b : lorem$</td></tr>
</table>

12.2. Állapottér transzformáció

Az alábbiakban egy olyan példát mutatunk be, amelyben az állapottér transzformáció nem egyszerű típustranszformáció. A feladatot úgy fogjuk megoldani, hogy eredeti állapottér helyett egy új (absztrakt) állapotteret választunk, azon megoldjuk a feladatot, majd a megoldásban szereplő műveleteket megvalósítjuk az eredeti téren.

Tegyük fel, hogy van egy karakterekből álló szekvenciális fi le-unk, ami egy szöveget tartalmaz. A feladat az, hogy számoljuk meg, hogy hány olyan szó van a szövegben, amelynek hossza nagyobb, mint a megadott k érték! A szövegben a szavakat elválasztó jelek (egy vagy több) választják el egymástól. Az elválasztó jelek halmazát jelölje S.

Így első ránézésre a feladat nem vezethető vissza egyetlen ismert programozási tételre sem. A feladatot ezért úgy általánosítjuk, hogy bevezetünk egy új állapotteret: tegyük fel, hogy a szövegfi le helyett egy olyan fi le-unk van, amiben az eredeti fi le szavainak hossza szerepel. Legyen ez az absztrakt fi le $g : G$, míg az eredeti szövegfi le $f : F$. A feladat specifi kációja az új téren:

$$A = \underset{g}{G} \times \underset{d}{\mathbb{Z}}$$
$$B = \underset{g'}{G}$$
$$Q : (g = g')$$
$$R : (d = \sum_{i=1}^{dom(g)} \chi(g_i > k))$$

Ez a feladat visszavezethető a számlálás tételének szekvenciális fi le-ra felírt változatára:

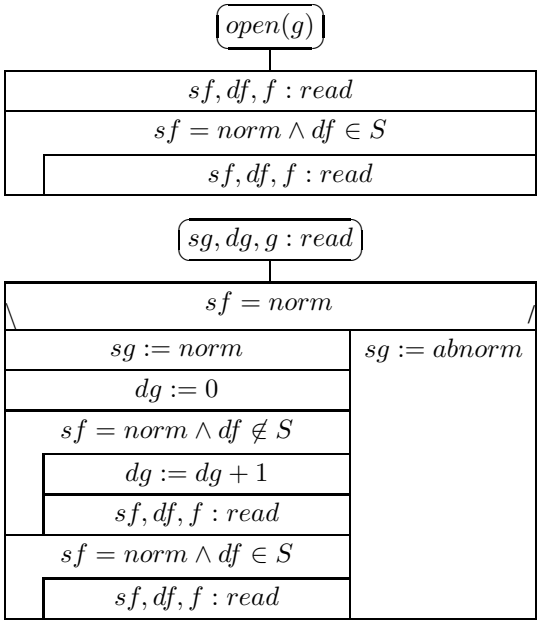
<table>
<tr><td colspan="2">$open(g)$</td></tr>
<tr><td colspan="2">$sg, dg, g : read$</td></tr>
<tr><td colspan="2">$d := 0$</td></tr>
<tr><td colspan="2">$sg = norm$</td></tr>
<tr><td colspan="2">$dg > k$</td></tr>
<tr><td>$d := d + 1$</td><td>$SKIP$</td></tr>
<tr><td colspan="2">$sg, dg, g : read$</td></tr>
</table>

Hátra van még az absztrakt $open$ és $read$ műveletek megvalósítása.

Az absztrakt g fi le-nak pontosan akkor van még eleme, ha az eredeti f fi le-ban van még szó. Ezért van szükség az $open$ műveletre, amely arra hivatott, hogy biztosítsa a $read$ művelet elején megkívánt invariáns tulajdonságot: vagy $sf = abnorm$ vagy df a következő szó első karakterét tartalmazza. Ezen invariáns tulajdonság segítségével a $read$ műveletben könnyen el lehet dönteni, hogy lesz-e visszaadott érték, vagy már az absztrakt fi le is kiürült.

Az absztrakt fi le-ok kezelésének általában is ez a technikája: egy $open$ művelettel biztosítjuk a $read$ elején megkívánt invariáns tulajdonságot, és gondoskodunk róla, hogy a $read$ művelet ezt az invariáns tulajdonságot megtartsa. Általában is igaz az, hogy az invariánsból egyszerűen eldönthető, hogy lesz-e még visszaadott elem, ezért *az absztrakt $read$ mindig elágazással kezdődik!*

Nézzük tehát az absztrakt műveletek megvalósítását az eredeti állapottéren: ezek mint látható szintén programozási tételek egyszerű konstrukciói.



12.3. Egyszerű programtranszformációk

Azt mondjuk, hogy az S program $p(S)$ programfüggvénye nem függ az állapottér A_i komponensétől, ha

$$\forall a, b \in \mathcal{D}_{p(S)} : (\forall k \in ([1,n] \setminus \{i\} : a_k = b_k) \rightarrow p(S)(a) = p(S)(b)).$$

Azt mondjuk, hogy az S program $a_i : A_i$ változója konstans, ha

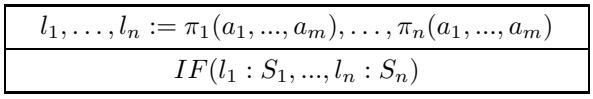
$$\forall a \in A : \forall \alpha \in S(a) : (\forall k \in \mathcal{D}_\alpha : \alpha_{ki} = a_i).$$

Azt mondjuk, hogy az S program végrehajtása nem változtatja meg az $a_i : A_i$ változót, ha

$$\forall a \in \mathcal{D}_{p(S)} : \forall b \in p(S)(a) : b_i = a_i$$

Nem megengedett feltétel kitranszformálása elágazásból. Legyen $S \subseteq A \times A^{**}, S = IF(\pi_1 : S_1, ..., \pi_n : S_n)$, $A = A_1 \times ... \times A_m$. Konstruáljuk az $S' \subseteq A' \times A'^{**}$ programot az alábbi módon:

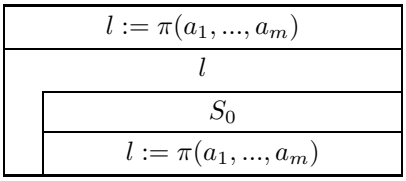
$$\underset{a_1}{A_1} \times \ldots \times \underset{a_m}{A_m} \times \underset{l_1}{\mathbb{L}} \times \ldots \times \underset{l_n}{\mathbb{L}} = A'$$



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

Nem megengedett ciklusfeltétel kitranszformálása. Let $S \subseteq A \times A^{**}, S = DO(\pi : S_0)$, $A = A_1 \times ... \times A_m$. Konstruáljuk az $S' \subseteq A' \times A'^{**}$ programot az alábbi módon:

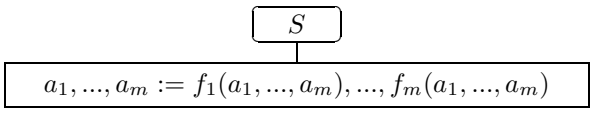
$$\underset{a_1}{A_1} \times \ldots \times \underset{a_m}{A_m} \times \underset{l}{\mathbb{L}} = A'$$



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

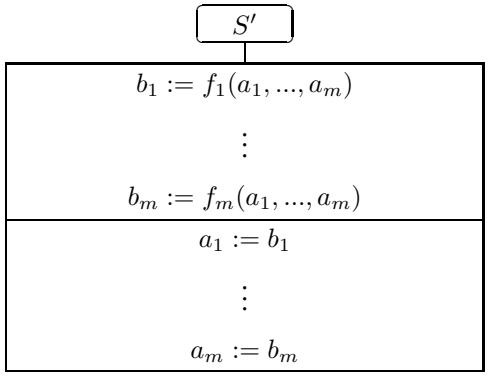
Szimultán értékadás helyettesítése egyszerű értékadásokkal. Legyen $S \subseteq A \times A^{**}$, a következő szimultán értékadás:

$$\underset{a_1}{A_1} \times \ldots \times \underset{a_m}{A_m} = A$$



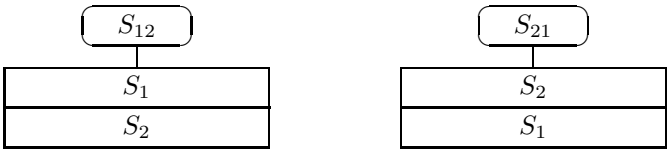
Konstruáljuk az $S' \subseteq A' \times A'^{**}$ programot az alábbi módon:

$$\underset{a_1}{A_1} \times \ldots \times \underset{a_m}{A_m} \times \underset{b_1}{A_1} \times \ldots \times \underset{b_m}{A_m} = A'$$

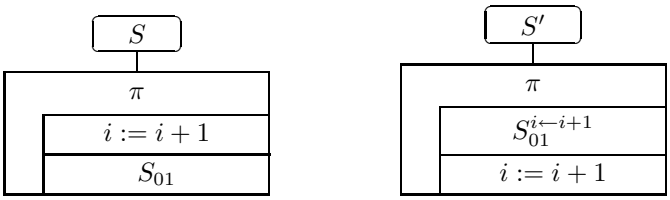


Ekkor az S' program ekvivalens S-sel A-n.

Szekvencia sorrendjének felcserélése. Ha $S_{12} = (S_1; S_2), S_{21} = (S_2; S_1)$ és az állapottér azon komponenseit amelyektől az S_1 függ S_2 nem változtatja meg és viszont, akkor S_{12} ekvivalens S_{21}-el.



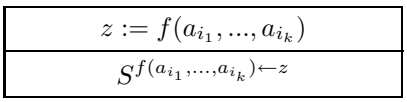
Ciklusmag-beli szekvencia sorrendjének felcserélése. Legyen $S = DO(\pi : S_0)$, $S_0 = (i := i + 1; S_{01})$ és tegyük fel, hogy az i konstans S_{01}-ben. Legyen továbbá $S' = ({S_{01}}^{i \leftarrow i+1}; i := i + 1)$.



Ekkor S ekvivalens S'-vel.

Függvény helyettesítése változóval. Legyen $S \subseteq A \times A^{**}$, $A = A_1 \times ... \times A_m$. és legyen f egy az $A_{i_1} \times ... \times A_{i_k}$ altér felett defi niált függvény, melynek értékkészlete H. Tegyük fel továbbá, hogy az $a_{i_1}, ..., a_{i_k}$ változók konstansok S-ben, és készítsük el az $S' \subseteq A' \times A'^{**}$ programot az alábbi módon:

$$A' = \underset{a_1}{A_1} \times \ldots \times \underset{a_m}{A_m} \times \underset{z}{H}$$



Ekkor S' ekvivalens S-sel A-n.

Rekurzívan definiált függvény helyettesítése. Legyen H egy tetszőleges halmaz, $k > 0$ egy egész szám, továbbá $F : \mathbb{Z} \times H^k \to H$ függvény, $t_0, t_{-1}, \ldots, t_{-k+1} \in \mathbb{Z}$ rögzített, és defi niáljuk az $f : \mathbb{Z} \to H$ függvényt az alábbi módon:

$$\begin{aligned} f(0) &= t_0, \\ f(-1) &= t_{-1}, \\ &\vdots \quad \vdots \\ f(-k+1) &= t_{-k+1} \end{aligned}$$

továbbá $\forall i \geq 0$:

$$f(i+1) = F(i+1, f(i), ..., f(i-k+1))$$

Tekintsük az alábbi S programot:

$i := 0$
S_1
π
S_0
$i := i + 1$

ahol i mind S_0-ban, mind S_1-ben konstans, és S_0-ban hivatkozás történik $f(i+1)$ értékére. Ekkor S ekvivalens az alábbi programmal:

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

13. fejezet

Szekvenciális megfelelő

Egy feladat megoldása során sokszor szembesülünk azzal a problémával, hogy különböző típusértékhalmazokba eső értékek közötti kapcsolatot kell leírnunk, vagy ilyen értékeket kell összehasonlítanunk. Erre leggyakrabban akkor kerül sor, ha valamilyen állapottértranszformációt végzünk, és meg kell adnunk az eredeti és az absztrakt tér közötti kapcsolatot. Az ilyen megfeleltetések formális megadása általában elég nehézkes. A *szekvenciális megfelelő* fogalmának felhasználásával azonban ezek a kapcsolatok is egyszerűbben írhatók fel.

Legyenek $E_1, E_2, \ldots, E_n$ elemi típusok (egy típus akkor elemi, ha önmagával van reprezentálva, azaz a hozzátartozó reprezentációs függvény az identikus leképezés). Az elemi értékek halmaza legyen

$$E = \bigcup_{i=1}^{n} E_i$$

Tegyük fel hogy a T típus reprezentációs függvényére igazak az alábbi megszorítások:

- $\varrho \subseteq E^* \times T$,
- ϱ kölcsönösen egyértelmű (bijektív),
- ϱ megengedett típuskonstrukció (azaz direktszorzat, unió és iterált konstrukciókból épül fel)

Legyen továbbá $B = \{B_1, B_2, \ldots, B_m\}$ az úgynevezett *bázishalmaz*, ahol B_i $(i = 1, \ldots, m)$ tetszőleges típusértékhalmaz, és jelölje az elemi típusok halmazát $F = \{E_1, E_2, \ldots, E_n\}$. Ekkor a $t \in T$ elem B bázisra vonatkoztatott szekvenciális megfelelője:

$$seq(t|B) = \begin{cases} \langle t \rangle, & \text{ha } T \in B \\ \langle\rangle, & \text{ha } T \notin B \wedge T \in F \\ \mathcal{R}(t|B), & \text{ha } T \notin B \wedge T \notin F \text{ és } T \text{ rekord} \\ \mathcal{E}(t|B), & \text{ha } T \notin B \wedge T \notin F \text{ és } T \text{ egyesítés} \\ \mathcal{S}(t|B), & \text{ha } T \notin B \wedge T \notin F \text{ és } T \text{ sorozat} \end{cases}$$

ahol

$$\begin{aligned} \mathcal{R}(t|B) &= kon(seq(t.s_1|B), \ldots, seq(t.s_k|B)) \\ \mathcal{E}(t|B) &= seq(\varphi_u^{-1}(t)|B) \\ \mathcal{S}(t|B) &= kon(seq(t_1|B), \ldots, seq(t_{dom(t)}|B)) \end{aligned}$$

A jelölés egyszerűsítése végett, ha a bázis egyelemű, akkor a bázishalmaz helyett az elem is írható, azaz $B = \{B_1\}$ esetén

$$seq(t|B_1) ::= seq(t|B).$$

Tekintsük a következő példát:

$$\begin{aligned} T &= seq(R), \\ R &= (nev : NEV, szul : SZUL), \\ NEV &= seq(CHAR), \\ SZUL &= (hely : HELY, ido : DATUM), \\ HELY &= seq(CHAR), \\ DATUM &= (ev : EV, ho : HO, nap : NAP), \\ EV &= \mathbb{N}_0 \\ HO &= \mathbb{N}_0 \\ NAP &= \mathbb{N}_0 \end{aligned}$$

Legyen $F = \{CHAR, \mathbb{N}_0\}$, $t \in T$. Ekkor

$seq(t\|NEV)$	a t sorozatbeli rekordok névrészeinek sorozata ($\in NEV^*$)
$seq(t\|CHAR)$	a t sorozatbeli rekordok névrészének és születési hely részének egymás után fűzésével kapott karaktersorozat
$seq(t\|\{HELY, EV\})$	a t sorozatbeli rekordok születési hely és év részének egymás után fűzésével kapott sorozat ($\in (HELY \cup EV)^*$)
$seq(seq(t\|NEV)\|CHAR)$	a t sorozatbeli rekordok névrészeiből képzett karaktersorozat
$seq(t\|\mathbb{R})$	üres sorozat

Hogyan használható a szekvenciális megfelelő az állapottértranszformáció leírására? Tekintsük a 12.2 fejezetben bemutatott példát. Ott egy szövegfile-t a benne levő szavak hosszainak file-jával helyettesítettük. Tegyük fel, hogy $CHAR = (ANUM; SEP)$ unió típus, ahol $ANUM$ a szavakat alkotó betűk típusa, SEP pedig az elválasztó jelek típusa. Ekkor az $f : F = seq(CHAR)$ eredeti file és a $g : G = seq(\mathbb{N})$ absztrakt file között kell a kapcsolatot definiálnunk. Ehhez vezessünk be még két absztrakt filetípust: legyen $U = seq(szo : SZO; szu : SZUNET)$, ahol $SZO = seq(ANUM)$ és $SZUNET = seq(SEP)$, továbbá legyen $V = seq(SZO)$. Ekkor:

- az $f : F$ és az $u : U$ közötti kapcsolat:

$$seq(u|\{ANUM, SEP\}) = seq(f|\{ANUM, SEP\}),$$

- az $u : U$ és a $v : V$ közötti kapcsolat:

$$seq(v|SZO) = seq(u|SZO),$$

- a $v : V$ és a $g : G$ közötti kapcsolat:

$$dom(g) = dom(v) \wedge \forall i \in [1..dom(g)] : g_i = dom(v_i).$$

Fel kell még tennünk azt, hogy u a lehető leghosszabb azonos típusú részsorozatokból áll, vagyis az U típus invariáns tulajdonsága:

$$I_U(u) = \forall i \in [1..dom(u) - 1] : u_i.szo \rightarrow u_{i+1}.szu \wedge u_i.szu \rightarrow u_{i+1}.szo$$

A fenti leírás matematikai egzaktsággal defi niálja azt az állapottértranszformációt, amit a 12.2 fejezetben csak szavakkal írtunk le. Természetesen másképp is formalizálható egy állapottértranszformáció, de a szekvenciális megfelelő használata sokszor leegyszerűsíti a leírást.

14. fejezet

Programinverzió

14.1. Egyváltozós eset

Legyenek A, B, C és D tetszőleges típusok. Legyen továbbá $X = seq(B)$ és $Y = seq(C)$, valamint $f_1 : A \to X$, $f_2 : X \to Y$ és $f_3 : Y \to D$ függvények.

Tekintsük az alábbi specifikációval adott feladatot:

$$\begin{array}{lll} A = & A & \times \ D \\ & a & \quad d \end{array}$$

$$\begin{array}{ll} B = & A \\ & a' \end{array}$$

$Q : (a = a')$

$R : (d = f_3 \circ f_2 \circ f_1(a'))$

Tegyük fel, hogy az f_1 függvény értékét kiszámító program az alábbi alakban írható fel:

$$\begin{array}{lll} A = & A & \times \ X \\ & a & \quad x \end{array}$$

$$\begin{array}{ll} B = & A \\ & a' \end{array}$$

$Q : (a = a')$

$R : (x = f_1(a'))$

$S_{11}(a)$	
$x :=<>$	
π	
	$S_{12}(a, e)$
	$x : hiext(e)$
$S_{13}(a)$	

Ha az S_{11}, S_{12} és S_{13} programok végrehajtása nem változtatja meg az x változó értékét, akkor a fenti programot *elemenként előállító* programnak nevezzük,

Hasonlóan, tegyük fel, hogy az f_3 függvény értékét kiszámító program pedig az alábbi formában írható fel:

$A = \underset{y}{Y} \times \underset{d}{D}$
$B = \underset{y'}{Y}$
$Q : (y = y')$
$R : (d = f_3(y'))$

$S_{31}(d)$
$y.dom \neq 0$
$S_{32}(d, y.lov)$
$y : lorem$
$S_{33}(d)$

Ha az S_{31}, S_{32} és S_{33} programok végrehajtása nem változtatja meg az y változó értékét, akkor a fenti programot *elemenként felhasználó* programnak nevezzük,

Tegyük fel továbbá, hogy az f_2 függvény elemenként feldolgozható, és minden egyelemű halmazra a függvényérték egyelemű. Ekkor a függvénykompozíció helyettesítési értékének kiszámítására vonatkozó programozási tétel alapján a feladat megoldható a fenti elemenként előállító, egy elemenként feldolgozó és az imént bemutatott elemenként felhasználó program szekvenciájaként.

A feladatra azonban egy hatékonyabb megoldást is adhatunk a *programinverzió* segítségével: ekkor a közbülső két sorozat típust kihagyhatjuk és a három ciklust egybeírhatjuk az alábbi módon:

$S_{11}(a)$
$S_{31}(d)$
π
$S_{12}(a, e)$
$\varepsilon := f_2(\{e\})$
$S_{32}(d, \varepsilon)$
$S_{13}(a)$
$S_{33}(d)$

14.2. Kétváltozós eset

Legyenek A^1, A^2, B, C és D tetszőleges típusok. Legyen továbbá $X = seq(B)$ és $Y = seq(C)$, valamint $f_1^1 : A^1 \to X$, $f_1^2 : A^2 \to X$, $f_2 : X \times X \to Y$ és $f_3 : Y \to D$ függvények.

Tekintsük az alábbi specifikációval adott feladatot:

$A = \underset{a^1}{A^1} \times \underset{a^2}{A^2} \times \underset{d}{D}$

$$B = \underset{a^{1\prime}}{A^1} \times \underset{a^{2\prime}}{A^2}$$

$Q : (a^1 = a^{1\prime} \wedge a^2 = a^{2\prime})$
$R : (d = f_3 \circ f_2(f_1^1(a^{1\prime}), f_1^2(a^{2\prime})))$

Legyenek az f_1^1 és f_1^2 függvényeket kiszámító elemenként előállító programok az alábbiak:

$S_{11}^1(a^1)$
$x^1 :=<>$
π^1
$S_{12}^1(a^1, e^1)$
$x^1 : hiext(e^1)$
$S_{13}^1(a^1)$

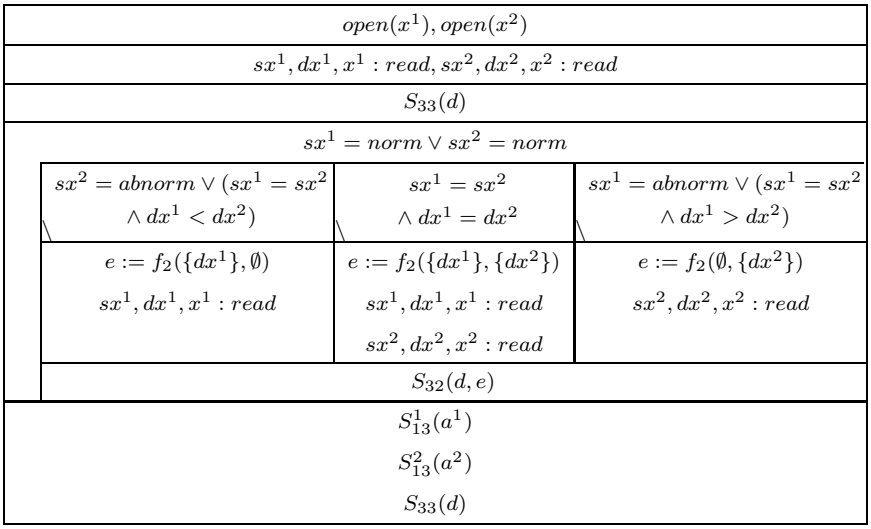
$S_{11}^2(a^2)$
$x^2 :=<>$
π^2
$S_{12}^2(a^2, e^2)$
$x^2 : hiext(e^2)$
$S_{13}^2(a^2)$

és tegyük fel, hogy az előállított x^1 és x^2 sorozatok rendezettek.

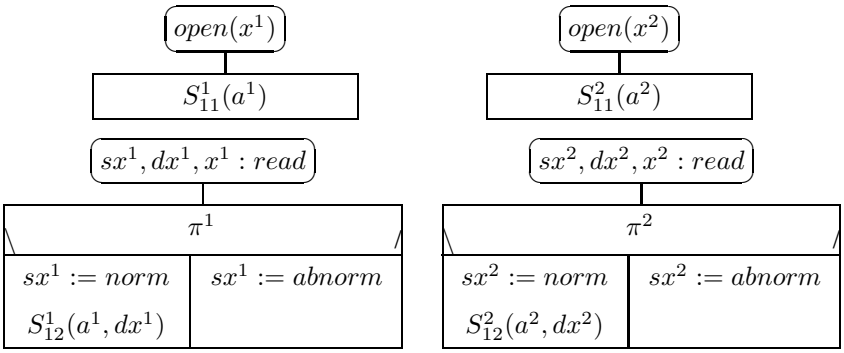
Legyen f_3 mint korábban, továbbá tegyük fel, hogy f_2 egy olyan kétváltozós egyértékű elemenként feldolgozható függvény, amely minden olyan halmazpárhoz, amelynek tagjai legfeljebb egy elemet tartalmaznak, pontosan egy elemű halmazt rendel hozzá.

Ekkor a függvénykompozíció helyettesítési értékének kiszámítására vonatkozó programozási tétel alapján a feladat megoldható a fenti két elemenként előállító, a kétváltozós egyértékű elemenként feldolgozó és az f_3-at kiszámító elemenként felhasználó program szekvenciájaként. Vajon ebben az esetben is összeinvertálhatóak a fenti programok?

A válasz természetesen: igen, de a megoldás itt nem olyan egyszerű, mint az egyváltozós esetben volt. A probléma a szekvenciális fi le-oknál felmerülttel analóg: az elemet előállító program (S_{12}^1 ill. S_{12}^1) az egyes elemeket ugyanúgy szolgáltatja mintha fi le-ból olvasnánkőket: csak akkor nézhetjük meg a következő elemet ha legeneráltatjuk. Ez sugallja azt a megoldást, hogy az x^1 és x^2 sorozatokat tekintsük az elemenként feldolgozásban absztrakt fi le-oknak. Az elemenként felhasználó program beinvertálása nem okoz gondot, ugyanúgy történik mint az egyváltozós esetben.



ahol az absztrakt műveletek megvalósításai:



15. fejezet

Időszerűsítés

15.1. Az időszerűsítés definíciója

Induljunk ki egy olyan adatfile-ból amelyben azonos típusú elemek találhatóak. Ezt a file-t *törzsfile*-nak fogjuk nevezni. Legyen az elemek típusa E, ekkor

$T = seq(E)$

Legyen adott továbbá egy olyan file, amely transzformációk sorozatát tartalmazza. Ezt a file-t fogjuk *módosítófile*-nak nevezni. Legyen $F = \{f \mid f : T \rightarrow T \text{ leképezés}\}$, ekkor

$M = seq(F)$

A feladat az, hogy időszerűsítsük a törzsfile-t a módosítófile-ban leírt transzformációkkal. Jelölje upd az időszerűsítés transzformációt. Ekkor $upd : T \times M \rightarrow T$ és

$$upd(t, m) = m_{dom(m)} \circ \cdots \circ m_2 \circ m_1(t)$$

Ahhoz, hogy a feladatra megoldást adjunk ez a leírás még mindig túl általános, ezért további kikötéseket teszünk a file-okra.

1. *Az időszerűsítés kulcsos*

   Legyen $E = (k : K, d : D)$ és $F = (k : K, v : V)$, ahol K egy tetszőleges rendezett halmaz, a rekordok kulcsrésze; D a törzsrekord adatrésze; V pedig az elvégzendő transzformációt definiáló típus. Feltesszük továbbá, hogy mind a tözsfile, mind a módosítófile a kulcsmező (k) szerint rendezett, azaz a T és M típusok invariáns tulajdonságára:

   $$\begin{aligned} I_T(t) &\Rightarrow \forall i \in [1..dom(t) - 1] : t_i.k \leq t_{i+1}.k \\ I_M(m) &\Rightarrow \forall i \in [1..dom(m) - 1] : m_i.k \leq m_{i+1}.k \end{aligned}$$

2. *A törzsfile a kulcsmező szerint egyértelmű*

   $$I_T(t) \Rightarrow \forall i, j \in [1..dom(t)], i \neq j : t_i.k \neq t_j.k$$

3. *A transzformáció csak az alábbi három féle lehet*

$$\begin{aligned} V &= (t : W_1; b : W_2; j : W_3) \\ W_1 &= \{\alpha\} \\ W_2 &= (d : D) \\ W_3 &= (g : G), \quad \text{ahol } G = \{\gamma \mid \gamma : D \to D\} \end{aligned}$$

Ahol a jelölés nem rontja el a szelektorfüggvények egyértelműségét, ott az egymás után következő szelektorfüggvény-sorozatból a közbülső elemek – a jelölést egyszerűsítendő – kihagyhatók, ezért pl. ha $dm : F$, akkor $dm.v.g$ helyett csak $dm.g$-t írunk.

Hátra van még annak leírása, hogy hogyan hatnak a fenti transzformációk a törzsfi le-ra. Kényelmi okokból a transzformációk eredményét halmazként adjuk meg (ez elegendő, mert a törzsfi le kulcs szerint egyértelmű). Ehhez bevezetünk néhány jelölést: legyen $x : seq(k : K, y : Y)$, ahol K rendezett halmaz, Y pedig tetszőleges típus, valamint $k \in K$. Ekkor

$$\begin{aligned} \{x\} &= \{x_i \mid i \in [1..dom(x)]\} \\ \{x.k\} &= \{x_i.k \mid i \in [1..dom(x)]\} \\ \mathcal{K}(x,k) &= \{x_i \mid i \in [1..dom(x)] \wedge x_i.k = k\} \end{aligned}$$

$$\{m_i(t)\} = \begin{cases} \mathcal{T}(m_i,t), & \text{ha } m_i.t \\ \mathcal{B}(m_i,t), & \text{ha } m_i.b \\ \mathcal{J}(m_i,t), & \text{ha } m_i.j \end{cases}$$

ahol

$$\mathcal{T}(m_i,t) = \begin{cases} \{t_j \mid t_j \in t \wedge t_j.k \neq m_i.k\}, & \text{ha } m_i.k \in \{t.k\} \\ \{t\}, & \text{különben} \end{cases}$$

$$\mathcal{B}(m_i,t) = \begin{cases} \{t\} \cup \{(m_i.k, m_i.v.d)\}, & \text{ha } m_i.k \notin \{t.k\} \\ \{t\}, & \text{különben} \end{cases}$$

$$\mathcal{J}(m_i,t) = \begin{cases} \{t_j \mid t_j \in t \wedge t_j.k \neq m_i.k\} \cup \\ \{(m_i.k, m_i.g(\mathcal{K}(t, m_i.k)))\}, & \text{ha } m_i.k \in \{t.k\} \\ \{t\}, & \text{különben} \end{cases}$$

4. *A javítás művelet csere*

$$I_M(m) \Rightarrow \forall i \in [1..dom(m)], m_i.v.j \Rightarrow m_i.g \text{ konstans}$$

Ebben az esetben a W_3 típust úgy szoktuk felírni, hogy a $g : G$ mező helyére $d : D$-t írunk és a $m_i.g(\mathcal{K}(t,k))$ függvényalkalkalmazást a módosítórekord adatrészére ($m_i.d$) cseréljük.

A feladat specifi kációja tehát:

$$\begin{array}{lccccc} A = & T & \times & M & \times & T \\ & t_0 & & m & & t \\ B = & T & \times & M & & \\ & t_0' & & m' & & \end{array}$$

Q : ($t_0 = t'_0 \wedge m = m'$ és az $1..x$ pontok teljesülnek)
R : ($t = upd(t'_0, m')$)

Ha a fenti specifikációban $x = 3$, akkor *közönséges*, ha $x = 4$ akkor *egyszerű* időszerűsítésről beszélünk.

15.2. Időszerűsítés egyértelmű módosítófile-lal

Mivel a közönséges és az egyszerű időszerűsítés között tulajdonképpen csak a javítás elvégzésében van különbség, mi a továbbiakban az egyszerű időszerűsítéssel fogunk foglalkozni. Közönséges időszerűsítés esetén az adatrész cseréjének helyére a javító függvény ($m_i.g$) elvégzése írható.

A feladatot három irányból is megpróbáljuk megoldani: visszavezetjük halmazok uniójára, egyváltozós egyértékű illetve kétváltozós egyértékű elemenkénti feldolgozásra.

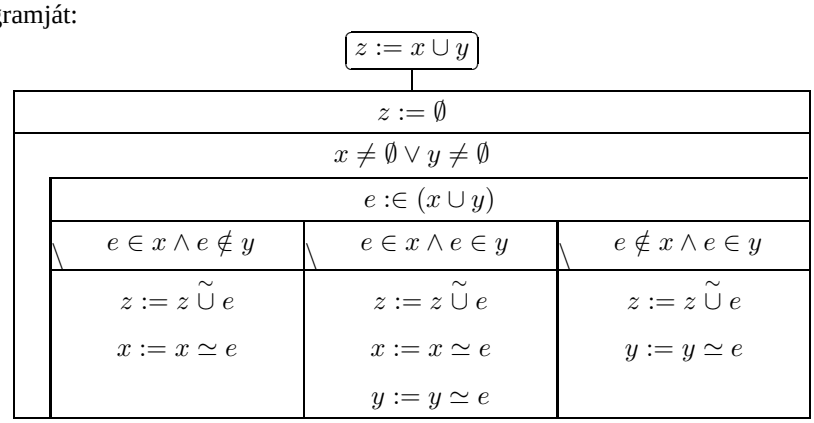
15.2.1. Visszavezetés halmazok uniójára

Tekintsük a file-okat halmaznak, és próbáljuk meg halmazokon megoldani a feladatot. Milyen kulcsértékek fordulhatnak elő az új törzsfile-ban? Természetesen csak olyanok, amelyek vagy t_0-ban, vagy m-ben, esetleg mindkettőben szerepeltek.

Terjesszük ki a D halmazt az üres értékkel: $D' = D \cup \{<\ddot{u}res>\}$ A D' segítségével az egyes transzformációk értelmezési tartománya és értékkészlete az alábbiak szerint alakul:

$$\begin{array}{ll} W_1: & D \rightarrow \{<\ddot{u}res>\} \\ W_2: & \{<\ddot{u}res>\} \rightarrow D \\ W_3: & D \rightarrow D \end{array}$$

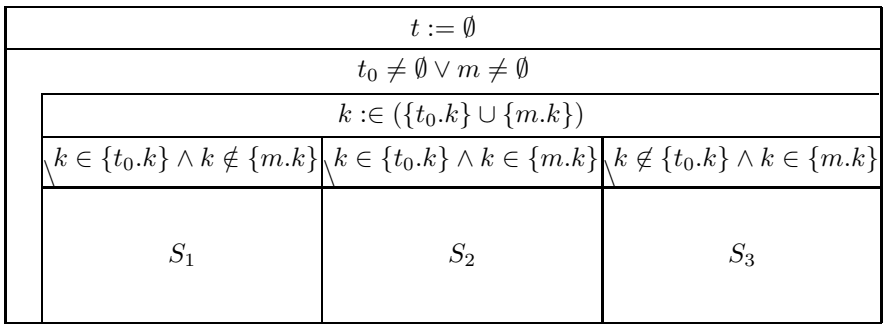
Ha egy elem adatrésze az $<\ddot{u}res>$ értéket veszi fel, akkor az azt jelzi, hogy valójában nem létezik, és ezért nem kerül bele az új törzsfile-ba. Idézzük fel az *unió* programját:



A feladat a következő megfeleltetéssel visszavezethető a fenti programra:

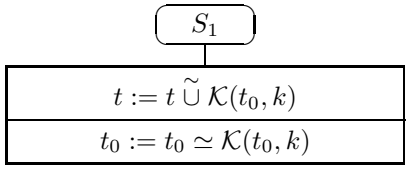
$$\begin{array}{rcl} x & \rightarrow & t_0 \\ y & \rightarrow & m \\ z & \rightarrow & t \\ x \neq \emptyset \vee y \neq \emptyset & \rightarrow & t_0 \neq \emptyset \vee m \neq \emptyset \\ e :\in (x \cup y) & \rightarrow & k :\in (\{t_0.k\} \cup \{m.k\}) \\ e \in x & \rightarrow & k \in \{t_0.k\} \\ e \in y & \rightarrow & k \in \{m.k\} \end{array}$$

A megfelelő program:

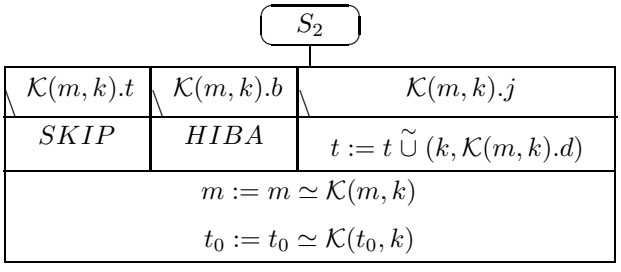


Vizsgáljuk meg most, hogy mit kell tenni az elágazás egyes ágaiban:

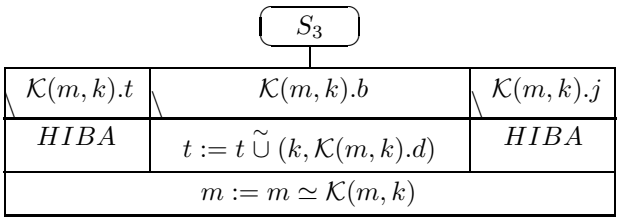
1. Ha a k kulcs csak a t_0 eredeti törzsfi le-ban szerepelt, akkor a $\mathcal{K}(t_0, k)$ elemre nem vonatkozott módosítás, változtatás nélkül kell kiírni az új törzsfi le-ba.



2. Ha a k kulcsérték mind az eredeti törzsfi le-ban, mind pedig a módosítófi le-ban szerepelt, akkor a törlés és a javítás műveletek végezhetők el. Ebben az ágban a k kulcsú elemet mindkét fi le-ból el kell hagyni.



3. Ha a k kulcs csak a módosítófi le-ban szerepelt, akkor csak a beszúrás műveletet lehet elvégezni, és a k kulcsú elemet ki kell törölni a módosítófi le-ból.



Ha a programnak hibajelzést is kell adnia, akkor azt a fenti struktogramokban $HIBA$-val jelzett helyeken kell megtennie. Térjünk most vissza az eredeti feladatra, ahol halmazok helyett szekvenciális fi le-ok szerepelnek. Legyen az inputfi le-okon a $read$ művelet értelmezve. Ekkor a program az alábbiak szerint alakul:

$$dx.d = \begin{cases} <\ddot{u}res>, & \text{ha } dx.k \notin \{t_0.k\} \\ \mathcal{K}(t_0, dx.k).d, & \text{ha } dx.k \in \{t_0.k\} \end{cases}$$

$$dx.v = \begin{cases} <\ddot{u}res>, & \text{ha } dx.k \notin \{m.k\} \\ \mathcal{K}(m, dx.k).v, & \text{ha } dx.k \in \{m.k\} \end{cases}$$

Az állapottértranszformációt alkalmazva így eljutottunk az

$$f(\{e\}) = \{(e.k, e.v(e.d))\}$$

elemenként feldolgozható függvényre felírt egyváltozós elemenkénti feldolgozáshoz, ahol a transzformációs rész alkalmazása az adatrészre:

$$e.v(e.d) = \begin{cases} e.d, & \text{ha } e.v = <\ddot{u}res> \\ <\ddot{u}res> & \text{ha } e.v.t \wedge e.d \neq <\ddot{u}res> \\ e.v.d & \text{ha } e.v.b \wedge e.d = <\ddot{u}res> \\ e.v.d & \text{ha } e.v.j \wedge e.d \neq <\ddot{u}res> \end{cases}$$

A specifi káció:

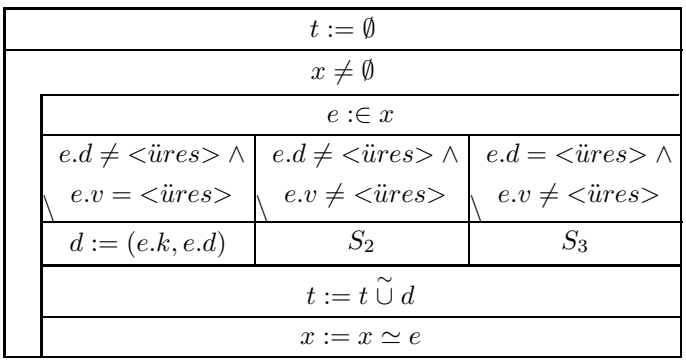
$$A = \underset{x}{X} \times \underset{t}{T}$$

$$B = \underset{x'}{X}$$

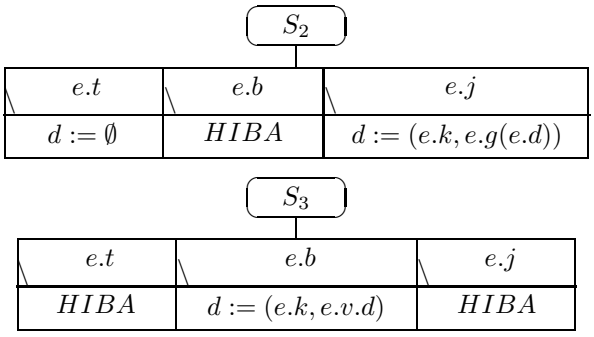
Q : ($x = x'$ és az $1..4$ pontok teljesülnek)

R : ($t = f(x')$)

A halmazokra felírt megoldóprogram:



ahol



Térjünk vissza most az eredeti állapottérre:

$x \neq \emptyset$	$\Rightarrow$	$t_0 \neq \emptyset \vee m \neq \emptyset$
$e :\in x$	$\Rightarrow$	$k :\in (\{t_0.k\} \cup \{m.k\})$
$e.d \neq <\ddot{u}res> \wedge e.v = <\ddot{u}res>$	$\Rightarrow$	$k \in \{t_0.k\} \wedge k \notin \{m.k\}$
$e.d = <\ddot{u}res> \wedge e.v \neq <\ddot{u}res>$	$\Rightarrow$	$k \notin \{t_0.k\} \wedge k \in \{m.k\}$
$e.d \neq <\ddot{u}res> \wedge e.v \neq <\ddot{u}res>$	$\Rightarrow$	$k \in \{t_0.k\} \wedge k \in \{m.k\}$

$x := x \simeq e$	$\Rightarrow$	$k \in \{t_0.k\}$ $\wedge k \notin \{m.k\}$	$k \in \{t_0.k\}$ $\wedge k \in \{m.k\}$	$k \notin \{t_0.k\}$ $\wedge k \in \{m.k\}$
		$t_0 := t_0 \simeq$ $\mathcal{K}(t_0, k)$	$t_0 := t_0 \simeq$ $\mathcal{K}(t_0, k)$ $m := m \simeq$ $\mathcal{K}(m, k)$	$m := m \simeq$ $\mathcal{K}(m, k)$

Használjuk fel azt a tényt, hogy a $d := f(\{e\})$ értékadást kiszámító programokban és az $x := x \simeq e$ értékadás megfelelőjében szereplő elágazások feltételrendszere megegyezik, továbbá a $t := t \mathbin{\tilde{\cup}} d$ értékadást csak azokba az ágakba írjuk bele, amelyekben $d \neq \emptyset$. Ekkor ugyanazt a programot kapjuk, mint az első megoldásban.

15.2.2. Visszavezetés kétváltozós elemenkénti feldolgozásra

A feladat megoldásának talán legegyszerűbb módja az, ha kétváltozós egyértékű – hibakezelés esetén kétértékű – elemenkénti feldolgozásra vezetjük vissza. Tekintsük a feladat eredeti specifikációját.

Ha a módosítófi le kulcs szerint egyértelmű, akkor az időszerűsítés függvénye (upd) a kulcsokra nézve elemenként feldolgozható. A kulcsértékekre felírt függvény:

$$
\begin{aligned}
upd(k, \emptyset) &= \mathcal{K}(t_0, k) \\
upd(\emptyset, k) &= \begin{cases} \emptyset, & \text{ha } \mathcal{K}(m,k).t \\ (k, \mathcal{K}(m,k).d), & \text{ha } \mathcal{K}(m,k).b \\ \emptyset, & \text{ha } \mathcal{K}(m,k).j \end{cases} \\
upd(k, k) &= \begin{cases} \emptyset, & \text{ha } \mathcal{K}(m,k).t \\ \mathcal{K}(t_0,k), & \text{ha } \mathcal{K}(m,k).b \\ (k, \mathcal{K}(m.k).g(\mathcal{K}(t_0,k).d)), & \text{ha } \mathcal{K}(m,k).j \end{cases}
\end{aligned}
$$

Ha ezt a fenti függvényt behelyettesítjük a kétváltozós elemenkénti feldolgozás tételébe, akkor ugyanahhoz a megoldóprogramhoz jutunk – csak sokkal rövidebb úton –, mint az első megoldásban.

15.3. Időszerűsítés nem egyértelmű módosítófile-lal

Vajon miben változik a feladat, ha a módosítófile kulcs szerint nem egyértelmű? Ebben az esetben a feladat nem elemenként feldolgozható. Erre a problémára kétféle megoldási módot is megvizsgálunk: az *adatabsztrakciós* és a *függvényabsztrakciós* megközelítést.

15.3.1. Megoldás adatabsztrakcióval

Mint az előbb már megállapítást nyert, ha a módosító file kulcs szerint nem egyértelmű, akkor a feladat nem elemenként feldolgozható. Hát akkor tegyük azzá! Ehhez arra van szükség, hogy a módosító file-t kulcs szerint egyértelművé tegyük. Ezt egy állapottértranszformáció segítségével könnyen megtehetjük, ugyanis csak annyit kell tennünk, hogy az azonos kulcsú módosító rekordokat egy új rekordba fogjuk össze. így az új módosító rekord a következőképpen fog kinézni:

(kulcs, transzformációsorozat)

Azaz definiáljuk az új módosítófile típusát az alábbi módon:

Legyen $W = seq(V)$; és $F' = (k : K, v : W)$. Ezekkel a típusdefiníciókkal a módosítófile így írható le:

$$X = seq(F');$$

Ezzel a módosítófile-lal tehát eljutottunk a kétváltozós egyértékű (hibafile használata esetén kétértékű) elemenkénti feldolgozáshoz. Az egyetlen különbség csak az, hogy az adott transzformáció-sorozat végrehajtását meg kell valósítanunk az eredeti állapottéren. Ehhez szükségünk lesz az $<üres>$ szibólumra, amely értéket hozzávesszük az adatrész típusához: $D' = D \cup \{<üres>\}$. Az, hogy egy törzsrekord adatrésze $<üres>$ azt jelenti, hogy a rekordot nem kell kiírni az eredményfile-ba.

Az elemenként feldolgozható függvényünk:

$$\begin{aligned} upd(k, \emptyset) &= \mathcal{K}(t_0, k) \\ upd(\emptyset, k) &= (k, \mathcal{K}(x, k).v(<üres>)) \\ upd(k, k) &= (k, \mathcal{K}(x, k).v(\mathcal{K}(t_0, k).d)) \end{aligned}$$

A transzformációsorozat elvégzése csak a transzformációk megfelelő sorrendje esetén lehetséges. Ha egy transzformációsorozat egy tagját nem lehet elvégezni, akkor azt a sorozatból ki kell hagyni (esetleg hibát kell jelezni). Ezzel az upd függvény egyértelműen definiált.

Írjuk fel tehát a fenti megfontolások alapján a kétváltozós elemenkénti feldolgozás programját a megfelelő behelyettesítésekkel:

<table>
<tr><td colspan="3">$t := 0$</td></tr>
<tr><td colspan="3">$st_0, dt_0, t_0 : read$</td></tr>
<tr><td colspan="3">$sx, dx, x : read$</td></tr>
<tr><td colspan="3">$st_0 = norm \vee sx = norm$</td></tr>
<tr><td>$sx = abnorm \vee (sx = st_0 \wedge dt_0.k < dx.k)$</td><td>$sx = st_0 \wedge dx.k = dt_0.k$</td><td>$st_0 = abnorm \vee (sx = st_0 \wedge dx.k < dt_0.k)$</td></tr>
<tr><td>$t : hiext(dt_0)$
$st_0, dt_0, t_0 : read$</td><td>$ak := dx.k$
$ad := dx.v(dt_0.d)$
$t : HIEXT(ad, ak)$
$st_0, dt_0, t_0 : read$
$sx, dx, x : read$</td><td>$ak := dx.k$
$ad := dx.v(<üres>)$
$t : HIEXT(ad, ak)$
$sx, dx, x : read$</td></tr>
</table>

Defi niálnunk kell még, hogy mit jelent a fenti struktogramban a transzformációsorozat elvégzése. Ehhez bevezetjük az $f : \mathbb{N}_0 \to D'$ rekurzívan defi niált függvényt:

$$dx.v(p) = f(dx.v.dom)$$

ahol

$$\begin{aligned} f(0) &= p \\ f(i+1) &= dx.v_{i+1}(f(i)) \end{aligned}$$

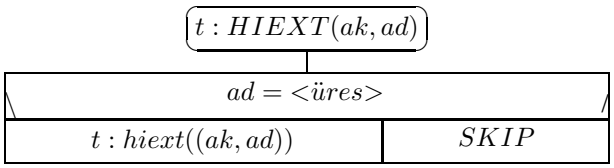
A fenti defi nícióban szereplő $dx.v_{i+1}$ művelet elvégzésén az alábbi függvényértékeket értjük: Legyen $d \in D'$, ekkor:

$$dx.v_{i+1}(d) = \begin{cases} <üres>, & \text{ha } dx.v.t \wedge d \neq <üres> \\ d, & \text{ha } dx.v.t \wedge d = <üres> \\ d, & \text{ha } dx.v.b \wedge d \neq <üres> \\ dx.v.d, & \text{ha } dx.v.b \wedge d = <üres> \\ dx.v.d, & \text{ha } dx.v.j \wedge d \neq <üres> \\ d, & \text{ha } dx.v.j \wedge d = <üres> \end{cases}$$

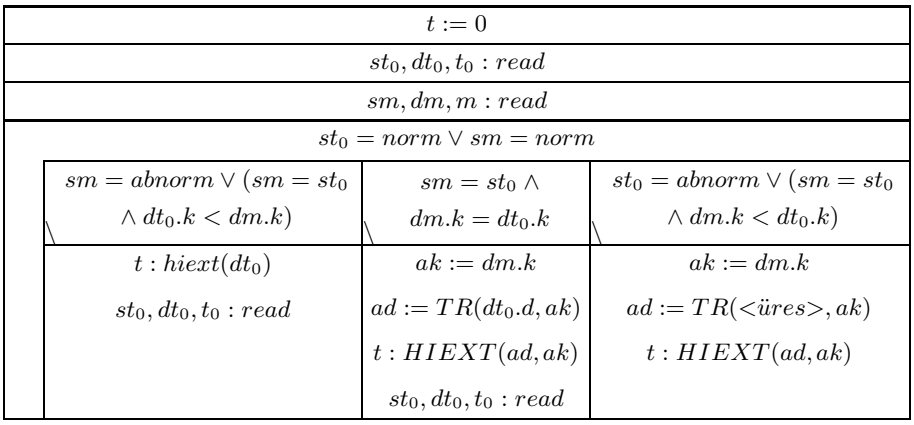
Az f függvényt kiszámító – a módosítássorozatot elvégző – program:

<table>
<tr><td colspan="6">$ad := dx.v(p)$</td></tr>
<tr><td colspan="6">$d := p$</td></tr>
<tr><td colspan="6">$dx.v.dom \neq 0$</td></tr>
<tr><td colspan="2">$dx.v.lov.t$</td><td colspan="2">$dx.v.lov.b$</td><td colspan="2">$dx.v.lov.j$</td></tr>
<tr><td colspan="2">$ad = <üres>$</td><td colspan="2">$ad = <üres>$</td><td colspan="2">$ad = <üres>$</td></tr>
<tr><td>$HIBA$</td><td>$ad := <üres>$</td><td>$ad := dx.v.lov.d$</td><td>$HIBA$</td><td>$HIBA$</td><td>$ad := dx.v.lov.d$</td></tr>
<tr><td colspan="6">$dx.v : lorem$</td></tr>
</table>

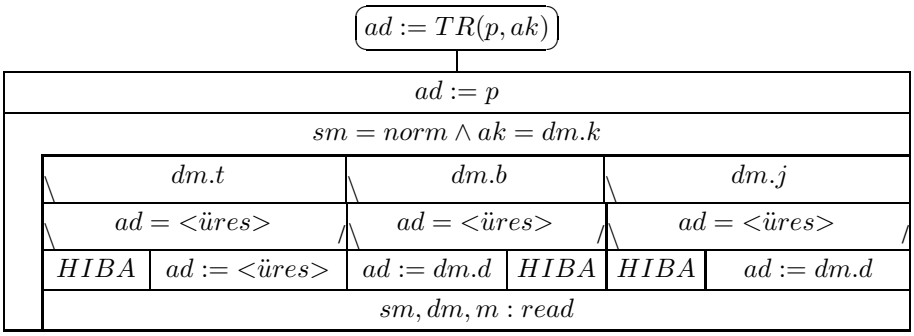
Mivel az aktuális adat értéke lehet $<üres>$ is, a $hiext$ művelet helyett az alábbi programot használjuk:



Térjünk most vissza az eredeti állapottérre. Ekkor a főprogram:



Az $ad := TR(p, ak)$ a már korábban definiált rekurzív függvényt kiszámító program megvalósítása az eredeti állapottéren:



15.3.2. Kulcsok egyértelműsítése

A gyakorlatban sokszor találkozhatunk olyan file-okkal, amelyek rekordjaiban van valamilyen kulcsmező ami szerint a file rendezett, ám de a file mégsem egyértelmű kulcs szerint, és így a file a kulcsmezőre vonatkoztatva nem elemenként feldolgozható. A következőkben egy olyan technikát fogunk bemutatni, amelyben egy új kulcsot definiálunk a file-ra, és ezen új kulcs szerint a file már egyértelmű.

Tegyük fel, hogy a file $U = (k : K, z : Z)$ típusú rekordokból áll. Az új kulcsot úgy kapjuk, hogy az egymás után levő azonos kulcsokat megsorszámozzuk. Legyen tehát $V = (h : H, z : Z)$, ahol $H = (k : K, s : \mathbb{N})$. Legyen továbbá $g : seq(U) \rightarrow seq(V)$:

1. $g(u).dom = u.dom$
2. $g(u)_1.s = 1$ és $\forall i \in [1..u.dom - 1]$:

$$g(u)_{i+1}.s = \begin{cases} 1, & \text{ha } u_i.k \neq u_{i+1}.k \\ g(u)_i.s + 1, & \text{ha } u_i.k = u_{i+1}.k \end{cases}$$

3. $\forall i \in [1..u.dom]$:
$$g(u)_i.k = u_i.k \text{ és } g(u)_i.z = u_i.z.$$

Ekkor tetszőleges $u \in seq(U)$ esetén – feltéve, hogy u a k kulcs szerint rendezett – $g(u)$ a h kulcs szerint rendezett és egyértelmű.

Természetesen a fenti megszámozást általában csak az absztrakció leírására használjuk, és csak ritkán fordul elő, hogy a g függvény által definiált absztrakciót meg is valósítjuk.

15.3.3. Megoldás függvényabsztrakcióval

Egy adott feladat megoldását mindig elkerülhetetlenül befolyásolja a specifikáció módja. A függvényabsztrakció lényege abban rejlik, hogy a megoldást egy alkalmasan választott függvény helyettesítési értékének kiszámítására vezetjük vissza.

Induljunk ki egy olyan absztrakt file-ból, mint amilyet az egyváltozós elemenkénti feldolgozásra való visszavezetésben használtunk. Természetesen, mivel most a módosítófile nem egyértelmű kulcs szerint, az X absztrakt file definíciója kissé módosul: ha egy kulcs mindkét file-ban szerepel, akkor a törzsrekordot az első rá vonatkozó módosítórekorddal vonjuk össze, és az esetlegesen előforduló további azonos kulcsú módosítórekordokból pedig egy-egy olyan absztrakt rekordot képezünk, amelynek adatrésze $<\ddot{u}res>$.

Ez az absztrakció az imént bemutatott egyértelműsítő leképezésen keresztül implicit módon írható le: legyen $t_0 \in T$, $m \in M$ és $x \in X$. Ekkor

$$\{g(x).h\} = \{g(t_0).h\} \cup \{g(m).h\}$$

és $\forall i \in [1..x.dom]$:

$$g(x)_i.d = \begin{cases} \mathcal{K}(g(t_0), g(x)_i.h).d, & \text{ha } g(x)_i.h \in \{g(t_0).h\} \\ <\ddot{u}res>, & \text{különben} \end{cases}$$

$$g(x)_i.v = \begin{cases} \mathcal{K}(g(m), g(x)_i.h).v, & \text{ha } g(x)_i.h \in \{g(m).h\} \\ <\ddot{u}res>, & \text{különben} \end{cases}$$

Megoldás függvénykompozícióval. Először egy olyan megoldást adunk a feladatra, amelynek specifikációjában az utófeltétel egy kompozícióval adott függvény kiszámítása, ahol a kompozíció egy rekurzív módon megadott függvényből és egy esetszétválasztással definiált függvényből áll.

$$A = \underset{x}{X} \times \underset{t}{T}$$
$$B = \underset{x'}{X}$$
$$Q : (x = x')$$
$$R : (t = HIEXT(f_1(x'.dom), (f_2(x'.dom), f_3(x'.dom))))$$

ahol $f : \mathbb{N}_0 \to T \times K \times D'$,

$$f(0) = (\langle\rangle, EXTR, <\ddot{u}res>)$$

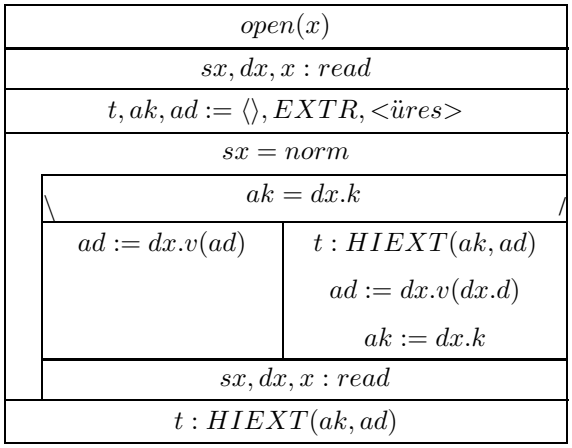
$$f(i+1) = \begin{cases} (f_1(i), f_2(i), x_{i+1}.v(f_3(i))), & \text{ha } x_{i+1}.k = f_2(i) \\ (HIEXT(f_1(i), (f_2(i), f_3(i))), \\ \quad x_{i+1}.k, x_{i+1}.v(x_{i+1}.d)), & \text{ha } x_{i+1}.k \neq f_2(i) \end{cases}$$

és $HIEXT : T \times (K \times D') \to T$,

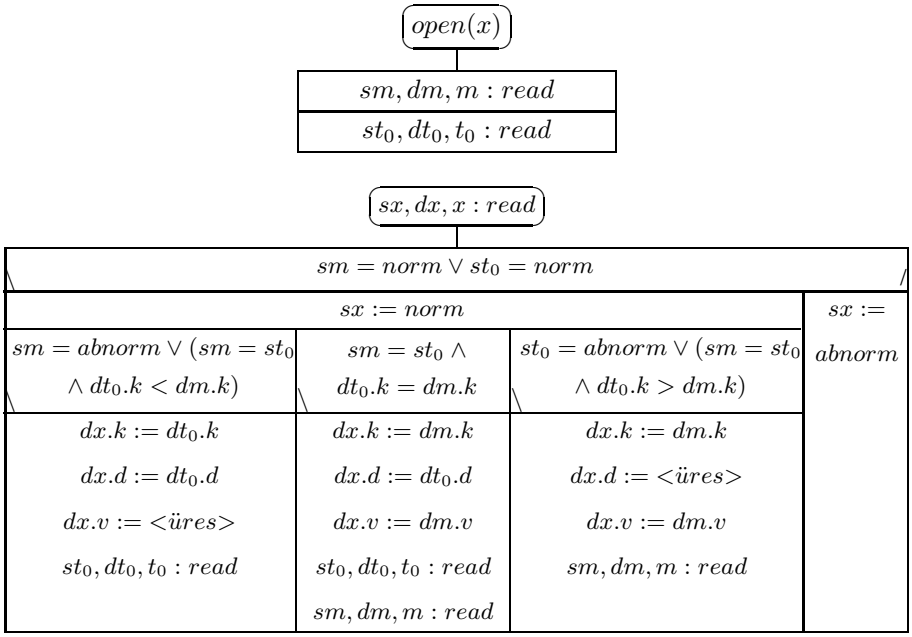
$$HIEXT(t,k,d) = \begin{cases} hiext(t,(k,d)), & \text{ha } d \neq <\ddot{u}res> \\ t, & \text{ha } d = <\ddot{u}res> \end{cases}$$

Az f függvény kezdőértékének definíciójában szereplő $EXTR$ kulcsérték egy tetszőleges olyan kulcsérték lehet, amely egyik file-ban sem fordul elő.

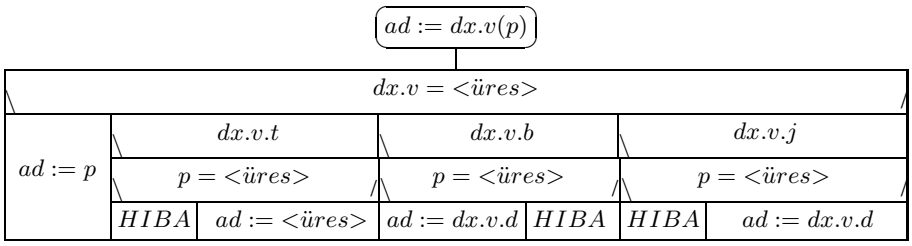
Mivel az utófeltétel függvénykompozícióval adott, a megoldás egy szekvencia lesz, amelynek első része az f, második része pedig a $HIEXT$ függvényt számítja ki. Az f függvény egyes komponenseinek rendre a t, ak, ad változók felelnek meg.



Az x absztrakt file műveleteinek megvalósítása:



Hátra van még a transzformáció elvégzésének megvalósítása:



Megoldás extremális elemmel. Az előző megodás szépséghibája, hogy a feladatot nem egy függvény helyettesítési értékének kiszámításaként specifikáltuk. Ezért adunk most egy másik megoldást is, amely egy a gyakorlatban sokszor hasznos eszközt használ. Ez az utolsó utáni elem bevezetése ($read$ extremális elemmel).

Egészítsük ki az előző megoldásban szereplő X file-t – ha nem üres – egy olyan elemmel, amelynek kulcsa minden lehetséges kulcsértéktől eltér. Jelöljük ezt a típust $\overline{X}$-sal. Ekkor a két típus közötti megfeleltetés: $\eta : X \to \overline{X}$,

$$\eta(x) = \begin{cases} hiext(x, (EXTR, <üres>, <üres>)), & \text{ha } x.dom \neq 0 \\ x, & \text{ha } x.dom = 0 \end{cases}$$

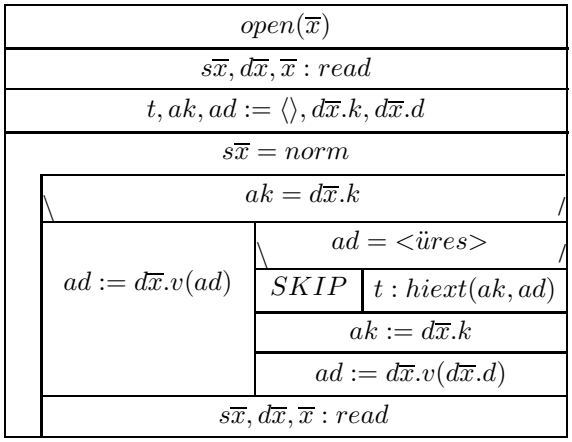
A kiszámolandó rekurzív függvény majdnem teljesen megegyezik az előzővel. A jobb áttekinthetőség érdekében a függvényt most komponensenként fogjuk felírni. Ahhoz, hogy a függvényt egyszerűbben írhassuk fel, tegyük fel, hogy az üres sorozatra alkalmazott lov függvény nem definiált értéket ad vissza (így a lov nem csak parciális függvény, hiszen minden sorozatra alkalmazható). Ekkor $f : \mathbb{N}_0 \to T \times K \times D'$, $f = (f_1, f_2, f_3)$

$$\begin{aligned}
f(0) &= (\langle\rangle, \overline{x}.lov.k, \overline{x}.lov.d) \\
f_1(i+1) &= \begin{cases} f_1(i), & \text{ha } f_2(i) = \overline{x}_{i+1}.k \vee \\ & \quad (f_2(i) \neq \overline{x}_{i+1}.k \wedge f_3(i) = <üres>) \\ hiext(f_1(i), (f_2(i), f_3(i))), & \text{ha } f_2(i) \neq \overline{x}_{i+1}.k \wedge f_3(i) \neq <üres> \end{cases} \\
f_3(i+1) &= \begin{cases} \overline{x}_{i+1}.v(f_3(i)), & \text{ha } f_2(i) = \overline{x}_{i+1}.k \\ \overline{x}_{i+1}.v(\overline{x}_{i+1}.d), & \text{ha } f_2(i) \neq \overline{x}_{i+1}.k \end{cases} \\
f_2(i+1) &= \begin{cases} f_2(i), & \text{ha } f_2(i) = \overline{x}_{i+1}.k \\ \overline{x}_{i+1}.k, & \text{ha } f_2(i) \neq \overline{x}_{i+1}.k \end{cases}
\end{aligned}$$

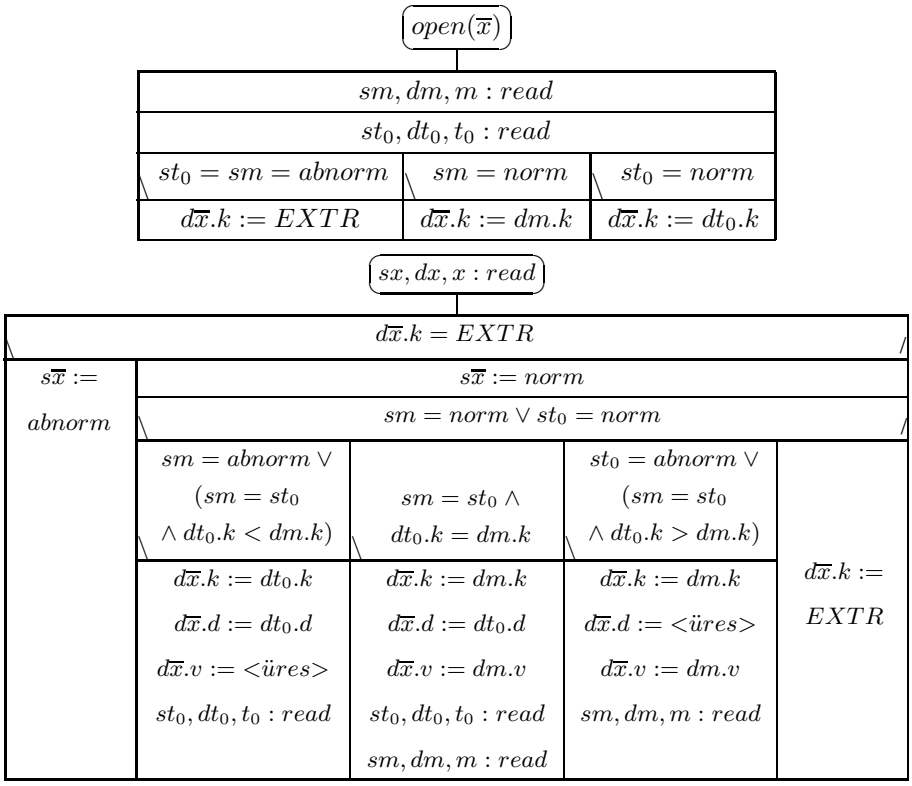
Ezt a függvényt használva a feladat specifikációja:

$$\begin{aligned}
A &= \overset{\overline{x}}{\overline{X}} \times \overset{t}{T} \\
B &= \overset{\overline{x}'}{\overline{X}} \\
Q &: (\overline{x} = \overline{x}') \\
R &: (t = f_1(\overline{x}'.dom))
\end{aligned}$$

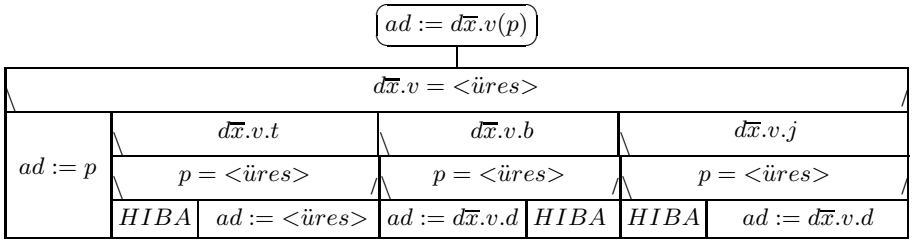
Ez a feladat visszavezethető file-ra felírt rekurzívan megadott függvény helyettesítési értékének kiszámítására:



Az $\overline{x}$ absztrakt fi le műveleteinek megvalósítása:



A transzformáció elvégzésének megvalósítása tulajdonképpen megegyezik az előző esetnél felírttal, csak most az $\overline{x}$ absztrakt fi le-t használjuk:



A fenti megoldási módokat összehasonlítva látható, hogy minél magasabb absztrakciós szintű fogalmakat használunk, annál egyszerűbben tudjuk kezelni a feladatot.

Nagyon sok esetben az adat- és függvényabsztrakció is alkalmazható egy feladat megoldásakor, sőt mint azt az iménti példa mutatja a kettő kombinációja is egy lehetséges út.