FÓTHI ÁKOS

BEVEZETÉS
A
PROGRAMOZÁSHOZ

ELTE Eötvös Kiadó

Lektor: Hunyadvári László

Nyelvileg lektorálta: Hunyady András

Tartalomjegyzék

Bevezetés

A könyvben ismertetett programozási modell megfogalmazásának elsődleges célja az volt, hogy elméleti hátteret adjon a lépésenkénti finomítás módszeréhez – az algoritmusok és az adatszerkezetek vonatkozásában egyaránt. Időközben ezen sikerült jelentősen túllépni, s a jelenlegi legfontosabb cél a tudatos programozás támogatása.

Bár a modell eszközei segítségével a feladat specifikációját helyettesíteni tudjuk olyan feladatok specifikációival, amely feladatok megoldása esetén a rendelkezésre álló matematikai eszközökkel belátható az eredeti feladat megoldásának helyessége, nem célunk az automatikus programszintézis, ezért nem fordultunk a formális eszközök, módszerek felé.

A programnyelvek, programok vizsgálata, helyességük bizonyítása már több mint negyven évre nyúlik vissza, sőt néhány fontos eredmény ennél is régebbi. Maga a programozás is nagyon hamar a vizsgálatok tárgyává vált, mind elméleti, mind gyakorlati szempontból. Itt csak utalunk a relációs modell számunkra legfontosabb előzményeire, és megpróbáljuk elhelyezni a véleményünk szerint legjelentősebb irányzatok között.

Előzmények

A bemutatott modell négy fontos előzményre épít. Dijkstra „programozási diszciplínájá”-ra [Dij 76], Hoare típuskezelésére [Hoa 72], Jackson programtervezési módszerére [Jac 75] és Mills matematikai megközelítésére [Mills 72].

A 60-as években kibontakozó, szoftverkrízisnek nevezett jelenség jelentős lendületet adott a programozással kapcsolatos kutatásoknak. Ezeknek szinte szimbólumává vált Dijkstra nevezetes cikke [Dij 68], és a programnyelvek által a kezdetek óta támogatott moduláris programozás mellett vagy inkább után a strukturált programozásnak nevezett irányzat kiinduló pontja lett.

Az általa bevezetett fogalmakra, módszerekre [Dij 76] nagyon sokan építettek, így a relációs modellünk is. Számunkra legfontosabbak a leggyengébb előfeltétel, a levezetési szabályok, a nemdeterminisztikusság, az állapottér, a Dikstra által használt, bár Hoare-tól származó [Hoa 72] elő- és utófeltétellel történő specifikáció.

Habár Dijkstra ebben a könyvében logikai jellegű megoldásokat használt, nem törekedett szigorú formalizálásra, ezt rövid időn belül nagyon sok cikk és könyv [Gri 81] megtette. A formalizálás legtöbbször logikai eszközökkel történt, aminek következtében háttérbe szorult az állapottér fogalma és a leggyengébb előfeltételt predikátum-transzformátorként kezelték.

Jelentősen befolyásolta a relációs modellünk kialakítását Mills [Mills 72] matematikai megközelítési módja. A programfüggvény fogalma, a függvény lezártjának és a ciklusnak a kapcsolata, a memória modellezése az, amit elsősorban figyelembe vettünk. A nemdeterminisztikusság kezelésére kézenfekvő megoldás volt a relációk használata.

A programozás problémáinak más jellegű megközelítését adta Jackson [Jac 75], ami a napjainkban félformálisnak nevezett módszerek egyik ősének tekinthető, és a gyakorlatban sokszor alkalmazták. Ezeket a programtervezési módszereket is igyekeztünk kezelhetővé tenni a relációs modellben.

Programozási paradigmák

A jelenlegi fő programozási irányzatok két nagy csoportba sorolhatók. Az egyikbe tartoznak a *formális módszerek*, amelyeknek az alapja egy-egy formális matematikai diszciplína.

A λ kalkuluson, illetve annak kiterjesztésein alapul a *funkcionális* paradigma. A nagy megbízhatóságú, bonyolult feladatokat megoldó programok iránti igény és a rendelkezésre álló erőforrások hihetetlen növekedése miatt a funkcionális megközelítés már nem elméleti, hanem elsősorban gyakorlati jelentőségű.

A *logikai alapú* programozás (modellezés) – különösen a logika különböző kiterjesztései, pl. temporális logika következtében – sok általános és speciális rendszer keretében jelenik meg.

Figyelemre méltó karriert fut be a *B-módszer* [Abr 96], ami Dijkstra és Hoare említett munkáin alapul. Az absztrakt gép fogalmára épül, és a Zermelo–Fraenkel axiomákon alapuló B nyelvet használja a rendszerkészítés minden fázisában.

A másik nagy csoportba tartoznak az úgynevezett *félformális rendszerek*. Ezek általában az objektumelvű modellalkotás támogatói, mint pl. az UML, de ide sorolhatjuk a *tervmintaalapú* tervezést [GHJV 95] is, ami a klasszikus eljáráskönyvtár általánosításának is tekinthető. Ezeknek a rendszereknek lényeges gyakorlati eleme a kód-újrafelhasználás, amit szintén az erőforrás-növekedés tesz lehetővé.

A relációs modellt az első csoporthoz a *levezetés*, a másodikhoz a *visszavezetés* fogalma kapcsolja.

A legújabb irányzatok közé tartoznak a *multiparadigmális* megközelítések [Bud 95], amikor a módszer alapja nem egyetlen absztraháló fogalom, illetve absztrakciós stratégia. Ebbe a csoportba soroljuk relációs modellünket is, de mi nem az eszközökkel, hanem a mögöttük levő elméleti összefüggésekkel foglalkozunk.

Az egyes fejezetekről

Az első részben összefoglaljuk a könyvben felhasznált legfontosabb matematikai fogalmakat, jelöléseket, és bevezetünk a relációkkal kapcsolatban néhány speciális fogalmat.

A következő két részben bevezetjük a relációalapú modell alapfogalmait: állapottér, feladat, program, megoldás. Megadjuk a megoldás egy elégséges feltételét, a gyakorlati szempontból is fontos specifikáció tételét, ehhez felhasználjuk a változó és a leggyengébb előfeltétel fogalmát.

A 4. és az 5. fejezetben általánosítjuk az eddig bevezetett fogalmakat. E fejezetek jelentősége elsősorban elméleti. Első olvasásra kihagyhatók.

A következő három részben a szokásos szekvenciális programkonstrukciókat vezetjük be, és megfogalmazunk néhány velük kapcsolatos, a lépésenkénti finomítás szempontjából hasznos tételt.

A 9. fejezet szintén elméleti szempontból érdekes, és nem szükséges a továbbiak megértéséhez.

Ezután a programozás legfontosabb fogalmával, a típussal foglalkozunk a 10. és a 11. fejezetben. A típussal kapcsolatos fogalmak nemcsak fontosak, hanem elég bonyolultak is. Első olvasásra nem is kell törekedni alapos megismerésükre. Célszerű az egész könyv végigolvasása után visszatérni ezekre a fejezetekre.

A következő három fejezetben azt mutatjuk meg, hogyan lehet gyakorlati programozási feladatokat megoldani. A 12. fejezetben a levezetési szabályok alkalmazásával, a feladatot lépésenként finomítva jutunk el a bizonyítottan helyes megoldó programhoz. A 13. fejezetben a visszavezetést mutatjuk be, azaz meglévő megoldásokat, tételeket használunk fel. A 14. fejezetben egy összetettebb feladatot oldunk meg különböző módokon.

Az utolsó részben a fejezetek végén szereplő feladatok megoldásához adunk útmutatót, sőt sok esetben megadunk egy lehetséges megoldást is.

Köszönetnyilvánítás

Ez a könyv az Eötvös Loránd Tudományegyetemen a programtervező matematikusok számára tartott előadásaim alapján készült. A több mint húsz éve megjelent jegyzethez képest sok minden változott. Ez alatt az idő alatt sok volt és jelenlegi kollégám vett részt e tárgy oktatásában, és segítette fejlődését. A húsz év alatt több ezer hallgatóval találkoztam ezen előadások kapcsán, az ő negatív és néha pozitív véleményük is sok segítséget jelentett nekem.

Jelenlegi és volt kollégáim: Fekete István, Gregorics Tibor, Horváth Zoltán, Konczné Nagy Márta, Kozics Sándor, Kozsik Tamás, Nyékyné Gaizler Judit, Sike Sándor, Steingart Ferenc, Szabóné Nacsa Rozália, Tőke Pál, Varga Zoltán, Vargyas Miklós, Venczel Tibor segítségéért köszönettel tartozom.

Külön köszönöm Steingart Ferencnek azt a nagy munkát, amit a könyv első elektronikus jegyzet változatának elkészítésével végzett, nélküle nem vágtam volna bele ennek a könyvnek a megírásába.

Köszönöm Bozsik József, Kovács Péter, Riskó Gergely és Sztupák Szilárd Zsolt hallgatóknak, hogy elkészítették a feladatok megoldását. Kovács Péter Bozsik József segítségével a 15.1.–15.8., Riskó Gergely a 15.9. és 15.11., Sztupák Szilárd Zsolt a 15.10. rész megoldásait készítette el.

Köszönöm Hunyadvári László gondos lektori munkáját, rengeteg hasznos észrevételét.

Ajánlás

Ajánlom e könyvet elsősorban a programtervező matematikus, programtervező informatikus hallgatóknak. Ajánlom azoknak is, akik programozással foglalkoznak, szeretik a kihívásokat és nyitottak. Ez a könyv nem könnyű olvasmány és nem is hasznos ismeretek, receptek gyűjteménye, de aki figyelmesen, gondolkodva elolvassa, valószínűleg más szemmel fogja látni az eddigi munkáját is.

A járt út biztonsága helyett ajánlom a járatlan út izgalmát és reményét.

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. Azt, hogy egy H halmaz véges, néha így is írjuk: $|H| < \infty$.

Egy H halmaz részhalmazainak halmazát, azaz a hatványhalmazát $\wp(H)$-val, véges részhalmazainak halmazát $\mathfrak{F}(H)$-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. Egy α 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 halmazára az A^{**} jelölést használjuk.

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, amelyet 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, amelyet úgy kapunk, hogy az eredeti sorozat minden azonos elemekből álló véges részsorozatát a részsorozat egyetlen elemével helyettesítjük. Egy $\alpha \in A^{**}$ sorozat redukáltját $red(\alpha)$-val 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

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

definícióval adott halmaz az A és B halmazok *direktszorzata*.

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

Az R reláció *értelmezési tartománya*:

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

a reláció *értékkészlete*:

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

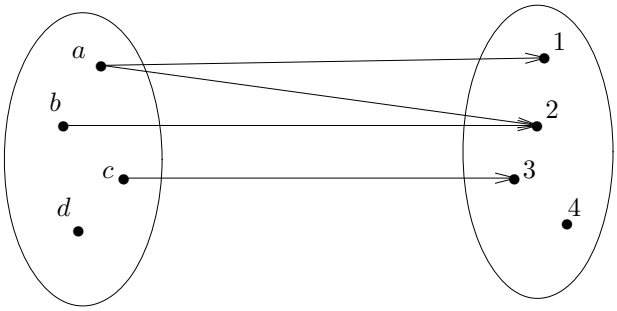
a reláció *értéke* egy $a \in A$ helyen, vagy másképpen a R szerinti *képe*:

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

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 nemdeterminisztikus leképezésnek, megfeleltetésnek is az A és a B halmaz elemei között. Az értelmezési tartomány jelenti azokat az A-beli elemeket, amelyekhez hozzárendelünk legalább egy B-beli elemet. Hasonlóan, az értékkészlet a legalább egy elemhez hozzárendelt elemek halmaza. Az 1.1. ábrával egy relációt szemléltetünk. $A = \{a, b, c, d\}, B = \{1, 2, 3, 4\}, R = \{(a, 1), (a, 2), (b, 2), (c, 3)\}$. Az a ponthoz hozzá van rendelve az 1 és a 2 is, b-hez



1.1. ábra. Reláció szemléltetése

csak a 2, c-hez a 3, d-hez nincs hozzárendelve semmi, és 4 nincs hozzárendelve semmihez.

Egy $a \in A$ elemnek a képe azoknak a B-beli elemeknek a halmaza, amelyeket a reláció hozzárendel az a-hoz. Egy H halmaz képét az elemek képeinek uniójával definiáltuk. Ez másképpen úgy fogalmazható meg, hogy a H képe azokból az elemekből áll, amelyek 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*, ha

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

A determinisztikus relációkat másképpen *parciális függvény*eknek hívjuk. *Függvény*nek nevezünk egy R relációt akkor, ha

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

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

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

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

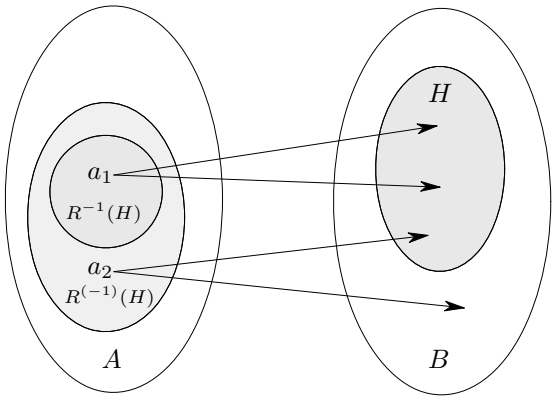
Legyen $H \subseteq B$ tetszőleges halmaz. H-nak az inverz reláció szerinti képét, $R^{(-1)}(H)$-t a H halmaz R reláció szerinti *inverz képének* nevezzük. A definíciókból látszik, hogy

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

Fontos megkülönböztetni az inverz képet a H halmaz R reláció szerinti *ősképé*től, amelynek 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 mindig része annak. A két kép kapcsolatát mutatja az 1.2. ábra. Megjegyezzük azonban, hogy függvény, illetve parciális függvény esetén a két kép megegyezik.



1.2. ábra. Inverz kép és őskép

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 $H \subseteq A$, akkor

$$R|_H ::= R \cap (H \times B)$$

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

1.3.1. Műveletek

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

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

és $Q \circ P$-vel jelöljük.

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

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

és $Q \odot P$-vel jelöljük.

Mint az az 1.3. 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 mindig része 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.

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

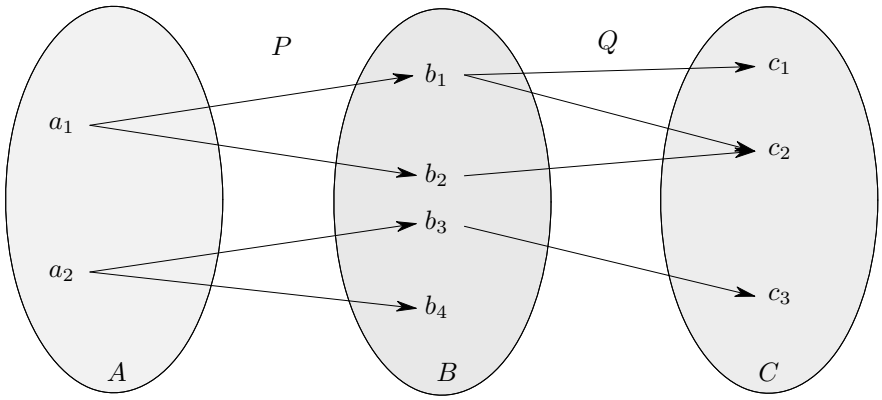
$$R^0 ::= \{(a, a) \mid a \in A\},$$

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:



1.3. ábra. Kompozíció és szigorú kompozíció

(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 $\overline{R}(a) = \{a\}$. 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 a gyenge igazsághalmaz megegyezik.

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

Az előzőekben definiáltuk két halmaz direktszorzatát, most ezt a fogalmat általánosítjuk.

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, hányadik komponens melyik $i \in I$-hez tartozik.

Megjegyezzük, hogy I üres is lehet, ebben az esetben a direktszorzat az üres halmaz.

Ha $J \subseteq I$ és $K = I \setminus J$, akkor 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.

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

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

Azt mondjuk, hogy az $A = \overset{\times}{\underset{i \in I}{}} A_i$ direktszorzat *ekvivalens* a $B = \overset{\times}{\underset{j \in J}{}} B_j$ direktszorzattal, ha létezik olyan $f : I \to J$ kölcsönösen egyértelmű megfeleltetés (bijekció), hogy $\forall i \in I : A_i = B_{f(i)}$. Úgy is fogalmazhatunk, hogy az értelmezési tartomány elemeinek átnevezésével kapjuk meg A-ból B-t. Könnyű belátni, hogy valóban ekvivalencia relációt definiáltunk a direktszorzatok között.

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 függvényt 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 $p, q : A \to \mathbb{L}$, ekkor:

$$\lceil p \wedge q \rceil = \lceil p \rceil \cap \lceil q \rceil,$$

$$\lceil p \vee q \rceil = \lceil p \rceil \cup \lceil q \rceil,$$

$$\lceil \neg p \rceil = A \setminus \lceil p \rceil,$$

$$\lceil p \to q \rceil = (A \setminus \lceil p \rceil) \cup \lceil q \rceil.$$

A $p = q$ jelölést, ha nem okoz félreértést az $p \equiv q$ értelmében is használjuk.

$$\lceil p \equiv q \rceil = \lceil p \rceil \cap \lceil q \rceil \cup \lceil \neg p \rceil \cap \lceil \neg q \rceil.$$

Az implikáció jelölésére gyakran használják a $\Rightarrow$ jelet a $\rightarrow$ helyett, mi az előbbit a „következik” reláció jelölésére használjuk, azaz

$$p \Rightarrow q \Leftrightarrow \lceil p \rceil \subseteq \lceil q \rceil.$$

Megjegyezzük, hogy $p \Rightarrow q$ és $p \rightarrow q = Igaz$ ugyanazt jelenti.

A $\forall$ és a $\exists$ jeleket eddig is használatuk mint a „minden” és „létezik” szavak rövidítéseit. 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 $0., 2.$ hatványa, mi R inverze?
d) Mi a $\{4, 5\}$ halmaz inverz képe, illetve ősképe?
e) Hány eleme van R értékkészlete hatványhalmazának?

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:

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

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 definíciója 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:

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

Az inverz kép definíciója alapján:

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

Az őskép definíciója alapján:

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

e) $|\wp(\mathcal{R}_R)| = 2^{|\mathcal{R}_R|} = 2^5 = 32$.

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

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

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 példán: 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 példán: 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

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

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

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

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 $\overline{R} = \{(4,5),(5,5)\}$ 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:

$\mathcal{D}_R = \mathbb{N}$. Minden 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 $\mathcal{D}_{\overline{R}} = \mathbb{N}_0$, 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. Legalább, illetve legfeljebb hány eleme van egy m elemű és egy n elemű halmaz

– metszetének; – egyesítésének;
– Descartes-szorzatának; – különbségének?

1.2. Bizonyítsa be, hogy $H \subseteq A \times B$ esetén

– $(\forall (a,b),(c,d) \in H : (a,d) \in H) \Leftrightarrow (\exists K \subseteq A : \exists L \subseteq B : H = K \times L)$;
– ha H nem üres, akkor K és L egyértelmű.

1.3. $R \subseteq A \times B$. Mivel egyenlő $R^{-1}(B)$?

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

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

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

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

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

1.9. Legyen $F, G \subseteq \mathbb{N} \times \mathbb{N}, Y = \{1,2\}$. $F = \{(a,b) \mid \ b|a$ és $b \neq 1$ és $b \neq a\}$.
$G = \{(a,b) \mid 2|a$ és $a = 2b\}$.

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

1.10. 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 és gyenge igazsághalmaza?

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

1.12. $F \subseteq A \times B, G \subseteq B \times C$. Igaz-e, hogy $\forall Y \subseteq C : (G \circ F)^{-1}(Y) = F^{-1}(G^{-1}(Y))$. Igaz-e az állítás, ha G vagy F függvény?

1.13. $F \subseteq A \times B, G \subseteq B \times C$. Igaz-e, hogy $(G \odot F)^{(-1)} = F^{(-1)} \odot G^{(-1)}$. Igaz-e az állítás, ha G vagy F függvény?

1.14. Mi az összefüggés két reláció kompozíciójának értelmezési tartománya és ugyanezen két reláció szigorú értelemben vett kompozíciójának értelmezési tartománya között?

1.15. Készítsen olyan nem üres R relációt és f logikai függvényt, hogy $f \circ R$ igazsághalmaza üres legyen!

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

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

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

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

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

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

1.19. $H \subseteq A \times B, G \subseteq B \times C, F \subseteq C \times D$. Igazak-e az alábbi állítások?

a) Ha $a \in \mathcal{D}_{G \circ H} \cap \mathcal{D}_{G \odot H}$, akkor $G \circ H(a) = G \odot H(a)$.

b) $\mathcal{D}_{G \odot H} \subseteq \mathcal{D}_{G \circ H}$.

c) $(\forall a \in \mathcal{D}_H : |H(a)| = 1) \Rightarrow G \circ H = G \odot H$.

d) $\mathcal{D}_G = B \Rightarrow G \circ H = G \odot H$.

1.20. Asszociativitás: $H \subseteq A \times B, G \subseteq B \times C, F \subseteq C \times D$. Igazak-e:

$$\begin{aligned} (F \circ G) \circ H &= F \circ (G \circ H), \\ (F \odot G) \odot H &= F \odot (G \odot H)? \end{aligned}$$

1.21. 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 nemszigorú kompozíció esetén?

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

$$\begin{aligned} R \subseteq S &\Rightarrow R \odot Q \subseteq S \odot Q, \\ R \subseteq S &\Rightarrow Q \odot R \subseteq Q \odot S? \end{aligned}$$

1.23. 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) Írja fel a két relációt, és addja meg az értelmezési tartományukat!

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

c) Írja fel a $Q \circ R$ relációt és annak értelmezési tartományát!

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

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

a) $\mathcal{D}_{G \odot F} = F^{-1}(\mathcal{D}_G)$,

b) $\mathcal{D}_{G \circ F} = F^{-1}(\mathcal{D}_G)$?

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

1.26. $R \subseteq \mathbb{N} \times \mathbb{N}$. $R = \{(a,b) \mid \ b|a$ és $b \neq 1$ és $b \neq a\}$. $\lceil\pi\rceil = \{x \mid \exists k \in \mathbb{N}_0 : x = 2^k\}$. Írjuk fel az $R|_\pi$ relációt, lezártját és korlátos lezártját!

1.27. 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!

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

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

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

Mi az R reláció lezártja és korlátos lezártja?

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

a) egy rögzített összetett természetes szám!

b) egy rögzített prímszám!

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

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

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

Mi lesz R lezártja és korlátos lezártja?

1.32. R legyen az 1.29. feladatban adott reláció. $\pi(k) = (k$ páratlan szám$)$. Adja meg az $R|_\pi$ relációt, lezártját és korlátos lezártját!

1.33. 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

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

1.34. Legyen $R \subseteq \mathbb{N}_0 \times \mathbb{N}_0$.

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

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

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

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

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

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

$$\begin{aligned}\alpha \quad = \quad & <(1,1,1,1),(1,2,1,1),(1,2,3,1),(1,2,3,4),\\ & (5,2,3,4),(5,7,3,4),(5,7,10,4),(5,7,10,14),\cdots>\end{aligned}$$

a) $pr_B(\alpha) =$?

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

2. fejezet

A programozás alapfogalmai

Ebben a fejezetben a programozás legfontosabb alapfogalmait vezetjük be. Nagyon fontos szempont, hogy figyelmünket 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 gondolkodásunk kiinduló pontja az, hogy kell lennie 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 megközelítésünk 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. Már csak 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 $A = \overset{\times}{_{i \in I}} A_i$ 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 –, amelyeket a modellünkben figyelembe akarunk venni.

Az állapottér fenti definíciójában 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 állnak. 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 pl. a 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 nevezünk egy $F \subseteq A \times A$ relációt.

A feladat fenti definíciója természetes módon adódik abból, hogy a feladatot egy leképezésnek tekintjük az állapottéren, és az állapottér minden pontjára megmondjuk, 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 derékszögű 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, amelyeket nem tudunk benne leírni, ha túl sok a komponens, akkor fölöslegesen túl bonyolult lesz a modell.

Tekintsük azt az egyszerű feladatot, hogy össze kell adni 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, 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, hogy „növeljünk 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 nemdeterminisztikusság 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 valódi osztóját! Ebben az esetben az állapottér is más lesz, hiszen egy természetes szám összes valódi osztója 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.

Egy program jellemző tulajdonsága, hogy „működik", azaz időben dinamikus folyamat. A dinamikus rendszerek általában nehezebben kezelhetők, 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 egymás utáni 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 előbbi 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 „filmre 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. $\forall a \in A : \forall \alpha \in S(a) : |\alpha| \neq 0$ és $\alpha_1 = a$.

A fenti definícióval 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, amelyekben nem szerepel egymás után ugyanaz az elem. Egész pontosan, ha ez mégis előfordul, akkor ez az elem ismétlődik végtelen sokszor. A működés abban nyilvánul meg, hogy megváltozik az állapot, vagy ha mégsem, az az abnormális működés jele. Emlékeztetünk arra, hogy az állapottér komponensei között minden előfordul, tehát ha bármi is történik, az más állapotérbeli pontot jelent. 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 definiáltuk, vagyis egy-egy állapottérbeli ponthoz több sorozat is hozzá lehet rendelve, azaz a működés nemdeterminisztikus. Ez első pillantásra talán meglepő, valójában természetes. Természetes akkor is, ha számítógépen, számítógéprendszeren futó programra gondolunk, hiszen egy program sok processzorból, sok, akár egymástól nagy távolságban levő komponensből álló rendszeren fut. De természetes akkor is, ha figyelembe vesszük, hogy a program-fogalom nemcsak 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 a benzin, mekkora a nyomás a kerekekben, működik-e az irányjelző, tevékenységek 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.

Az előbbiek miatt bevezetjük 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, ahonnan 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.

Megjegyezzük, hogy a programfüggvényen túl természetesen vannak a programnak olyan jellemzői, melyek a program minőségére vonatkoznak, és ezek 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.

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.

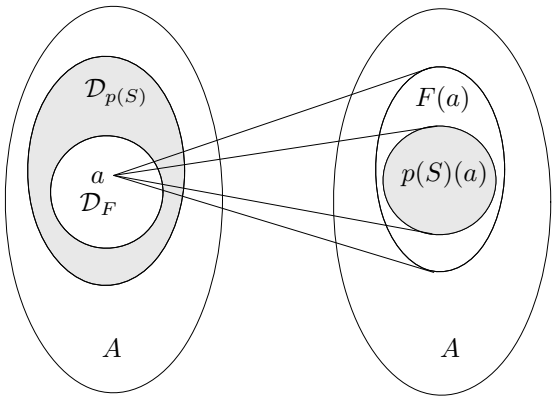
2.5. DEFINÍCIÓ: MEGOLDÁS

Azt mondjuk, hogy az S program *megoldja* az F feladatot, ha

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

Ezzel a definícióval végül is azt kívánjuk meg, hogy az állapottér olyan pontjaihoz, ahol a feladat értelmezve van, a program csak véges sorozatokat rendeljen (terminálj on), 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 megoldást!”



2.1. ábra. Megoldás

Pedig elég csak az autós példára gondolni. Például a fékfolyadék szintjének a minimum és a maximum szint jelzése között kell lennie. Ezt a karbantartás eredményeképpen teljesítjük, de egyáltalán nem egyértelmű, 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 definícióját összevetve a megoldás definíciójával könnyen adódik a következő egyszerű, de fontos állítás:

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 megoldóprogram? 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 $\forall a \notin \mathcal{D}_F : S(a) = \{\langle a, \ldots\rangle\}$. Nyilvánvaló, hogy $p(S) = F$, tehát S megoldása F-nek. Vagyis tetszőleges feladathoz könnyen tudunk megoldóprogramot csinálni. Ebből kiindulva azt gondolhatnánk, hogy a programozás nagyon egyszerű feladat. Ez persze nem igaz. A programozás feladata valójában sokkal összetettebb, mert egy programot adott programokból, rögzített szabályok szerint kell összeraknunk, például egy programnyelv eszközeit kell használnunk.

2.7. DEFINÍCIÓ: PROGRAMOZÁSI FELADAT

Legyen $A = \overset{\times}{_{i \in I}} A_i$. *Programozási feladatnak* nevezzük az $(F, \mathbb{P}, \mathbb{K})$ hármast, ahol $F \subseteq A \times A$ egy feladat; $\mathbb{P}$ a *primitív* (megengedett) programok véges

halmaza, $\forall S \in \mathbb{P} : S \subseteq A \times A^{**}$; $\mathbb{K}$ a megengedett programkonstrukciók véges halmaza, $\forall K \in \mathbb{K}$, egy az A-n értelmezett programok halmazán értelmezett művelet.

2.8. DEFINÍCIÓ: PROGRAMOZÁSI FELADAT MEGOLDÁSA

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

A programozási feladat két értelemben is általánosítható: egyrészt kibővíthető a program működésére vonatkozó feltételekkel, ezzel ebben a könyvben nem foglalkozunk. Másrészt nem követeljük meg az azonos állapotteret, ehhez általánosítjuk a megoldás fogalmát, illetve bevezetjük a típusspecifikáció, típus, megfelelés és a típuskonstrukciók fogalmát.

Az, hogy milyen programkonstrukciókat engedünk meg, sok mindentől 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 megoldóprogramot, hanem a feladatot bontjuk fel részfeladatokra, 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:

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

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

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

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 e 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 definíciójának 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

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

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 feladat. Tegyük 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 programfüggvényükről? Nézzük először az értelmezési tartományokat! A definíció 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 $A = \{\Omega, \Phi, \Psi, \Theta, \Gamma\}, S \subseteq A \times A^{**}$.

$$\begin{array}{lllll} S = \{ & (\Omega, \langle \Omega\Phi\Gamma\Omega \rangle), & (\Omega, \langle \Omega\Theta\Psi\Gamma \rangle), & (\Omega, \langle \Omega\Psi\Phi \dots \rangle), & \\ & (\Phi, \langle \Phi\Omega \rangle), & (\Psi, \langle \Psi\Theta \rangle), & (\Psi, \langle \Psi\Psi\Psi\Psi\Psi \dots \rangle), & \\ & (\Theta, \langle \Theta\Omega\Gamma\Omega\Theta \rangle), & (\Theta, \langle \Theta\Psi\Omega\Phi\Gamma\Omega \rangle), & (\Theta, \langle \Theta\Omega\Gamma\Theta\Phi \rangle), & \\ & (\Gamma, \langle \Gamma\Phi\Psi \rangle), & (\Gamma, \langle \Gamma\Psi \rangle), & (\Gamma, \langle \Gamma\Phi\Psi\Omega \rangle) & \} \end{array}$$

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

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

b) Megoldja-e S az F feladatot?

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

a) ha F nemdeterminisztikus, akkor S sem az?

b) ha F determinisztikus, akkor S is az?

c) ha F nemdeterminisztikus, 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 nemdeterminisztikus, 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:

$$q \in \mathcal{D}_F \Rightarrow S(q) \subseteq A^* \text{ és } p(S)(q) \subseteq F(q)?$$

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

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

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

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 ha $a \in A$, akkor

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

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, 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 az $lf(S,R) : A \to \mathbb{L}$ 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, ahonnan 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, hogy 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

3.1. állítás:

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

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

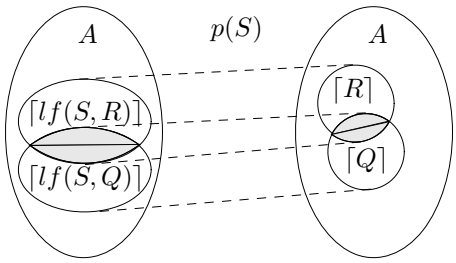
Az első tulajdonságot a csoda kizárása elvének, a másodikat monotonitási tulajdonságnak nevezzük.

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 $\lceil Q\rceil \subseteq \lceil R\rceil$

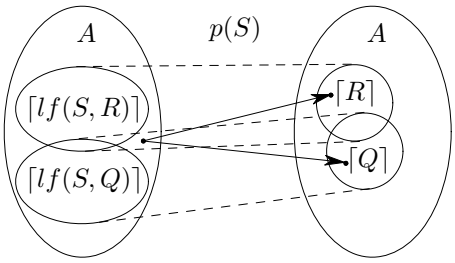
3. Az állítást két részben, a mindkét irányú következés belátásával bizonyítjuk.



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) $lf(S, Q \wedge R) \Rightarrow lf(S,Q) \wedge lf(S,R)$, ugyanis:
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 tulajdonság visszafelé nem igaz. $p(S)(a) \subseteq \lceil Q \rceil \cup \lceil R \rceil$-ből 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. $\forall a \in \mathcal{D}_F : p(S)(a) \subseteq F(a)$, ugyanis:

   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 elmondottak 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 $A = \overset{\times}{_{i \in I}} A_i$ 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 tartozó 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 változónevek 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 tételének 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, miért éppen így választottuk meg a paraméterteret.

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 felhasználásával 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, A-n értelmezett 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 mindegyik 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 nyilvánvaló, hogy a paraméterváltozók argumentuma b, ezek el is hagyhatók, sőt általában az sem okoz félreértést, ha az elő- és 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 arra, 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, amelyek *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.

A változókhoz kapcsolódóan bevezetünk még néhány egyszerű fogalmat, amelyeknek a szemléletes jelentése eléggé nyilvánvaló.

Azt mondjuk, hogy az S program $p(S)$ programfüggvénye nem függ az állapottér $a_i : A_i$ változójától, ha

$$\forall x, y \in \mathcal{D}_{p(S)} : (\forall k \in ([1, n] \setminus \{i\} : x_k = y_k) \rightarrow p(S)(x) = p(S)(b)).$$

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

$$\forall a \in \mathcal{D}_{p(S)} : \forall b \in p(S)(a) : b_i = a_i.$$

3.4. Példák

3.1. példa: Legyen $A = \{Bach, Bartók, Kodály, Liszt, Mozart, Vivaldi\}$, $S \subseteq A \times A^{**}$ program.

$$\begin{array}{lllll} S = \{ & Vivaldi & \to \langle Vivaldi, Bach\rangle, & Bach & \to \langle Bach, Mozart\rangle, \\ & Bach & \to \langle Bach, Liszt, Bartók\rangle, & Mozart & \to \langle Mozart, Vivaldi\rangle, \\ & Liszt & \to \langle Liszt, Bartók\rangle, & Kodály & \to \langle Kodály, Mozart\rangle, \\ & Bartók & \to \langle Bartók, Bach, Liszt\rangle\} & & \end{array}$$

Legyen továbbá az $R : A \to \mathbb{L}$ állítás:
$\forall x \in A: \;\; R(x) = (x$ magyar$)$.
Mi lesz a fenti program R-hez tartozó leggyengébb előfeltétele?
Megoldás: Írjuk fel először a program programfüggvényét:

$$\begin{array}{lll} p(S) = \{ & (Vivaldi, Bach), & (Bach, Mozart), \quad (Bach, Bartók), \\ & (Mozart, Vivaldi), & (Liszt, Bartók), \quad (Kodály, Mozart), \\ & (Bartók, Liszt) & \} \end{array}$$

Ezek után, a leggyengébb előfeltétel definícióját felhasználva:
$\lceil lf(S, R)\rceil = \{Bartók, Liszt\}$, ugyanis

$$\begin{aligned} p(S)(Vivaldi) &= \{Bach\} \not\subseteq \lceil R\rceil \\ p(S)(Kodály) &= \{Mozart\} \not\subseteq \lceil R\rceil \\ p(S)(Bartók) &= \{Liszt\} \subseteq \lceil R\rceil \\ p(S)(Bach) &= \{Mozart, Bartók\} \not\subseteq \lceil R\rceil \\ p(S)(Mozart) &= \{Vivaldi\} \not\subseteq \lceil R\rceil \\ p(S)(Liszt) &= \{Bartók\} \subseteq \lceil R\rceil \end{aligned}$$

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 belátható, 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: $A = \mathbb{L} \times \mathbb{L}, \quad F \subseteq A \times A$,

$$F = \{((l, k), (m, n)) \mid n = k \wedge m = (l \wedge k)\}.$$

Megoldás:

$$\begin{array}{lll} A = & \mathbb{L} & \times \; \mathbb{L} \\ & x & \quad y \end{array}$$

$$B = \underset{x'}{\mathbb{L}} \times \underset{y'}{\mathbb{L}}$$
$$Q : (x = x' \wedge y = y')$$
$$R : (x = (x' \wedge y') \wedge y = y')$$

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 $\forall b \in B$:

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

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 definíciójának 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 (ugyanis „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 $A = \{1, 2, 3, 4, 5\}, S \subseteq A \times A^{**}$.

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

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

3.2. Mivel egyenlő $lf(S, Igaz)$?

3.3. Legyen A tetszőleges állapottér, $Q_i : A \to \mathbb{L} \quad (i \in \mathbb{N})$. Igaz-e, hogy 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, hogy 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, hogy 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. $A = \mathbb{N}$. $S \subseteq \mathbb{N} \times \mathbb{N}^{**}$.

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

$H(x) = (x$ 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:

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

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

$Q : (x = x')$

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

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

Megadható-e valamilyen összefüggés F_1 és F_2 között?

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

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

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

$Q : (m = m' \wedge n = n' \wedge m \leq n)$,

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

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

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

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:

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

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

$Q : (k = k' \wedge 0 < k)$

$R : (Q \wedge prim(p) \wedge \forall i > 1 : prim(i) \rightarrow |k - i| \geq |k - p|)$,

ahol $prim(x) = (x$ prímszám$)$.

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

3.13.

$$\begin{array}{lccccc} A = & \mathbb{N} & \times & \mathbb{N} & \times & \mathbb{N} \\ & x & & y & & z \end{array}$$

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

$F_1, F_2 \subseteq A \times A$

F_1 specifikációja:

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

$R : (x = x' \wedge y = y' \wedge x'|z \wedge y'|z \wedge \forall j \in \mathbb{N} :\ (x'|j \wedge y'|j) \rightarrow z|j)$

$F_2 = \{((a, b, c), (d, e, f)) \mid a = d$ és $b = e$ és $f|a \cdot b$ és $a|f$ és $b|f\}$

Megadható-e valamilyen összefüggés F_1 és F_2 között?

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

$$\begin{array}{lccccc} A = & \mathbb{Z} & \times & \mathbb{Z} & \times & \mathbb{Z} \\ & m & & n & & i \end{array}$$

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

$F_1, F_2 \subseteq A \times A$

F_1 specifikációja:

$Q :(m = m' \wedge n = n')$

$R :(m = m' \wedge n = n' \wedge i \in [m..n] \wedge \forall j \in [m..i-1] :\ f(j) < f(i) \wedge$
$\forall j \in [i..n] :\ f(j) \leq f(i))$

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 $v : \mathbb{N} \rightarrow \{0, 1\}$.

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

3.16. Keressük meg egy természetes szám egy osztóját.

3.17. Keressük meg egy összetett természetes szám egy valódi osztóját.

3.18. Keressük meg egy természetes szám egy valódi osztóját.

3.19. Keressük meg egy természetes szám összes valódi osztóját.

3.20. Keressük meg egy természetes szám legnagyobb prímosztóját.

3.21. Állapítsuk meg, hány valódi osztója van egy természetes számnak.

3.22. Keressük az $[m..n]$ intervallumban az első olyan számot, amelyiknek van valódi osztója.

3.23. Keressük az $[m..n]$ intervallumban azt a számot, amelyiknek a legtöbb valódi osztója van, de nem osztható 6-tal.

3.24. Az $[m..n]$ intervallumban melyik számnak van a legtöbb valódi osztója?

4. fejezet

Kiterjesztések

Az előző fejezetekben bevezettük a program és a feladatat fogalmát, és definiáltuk az azonos állapottéren levő feladat-program párok között a megoldás fogalmát. A gyakorlatban azonban általában a feladat és a program különböző állapottéren van értelmezve; 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

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

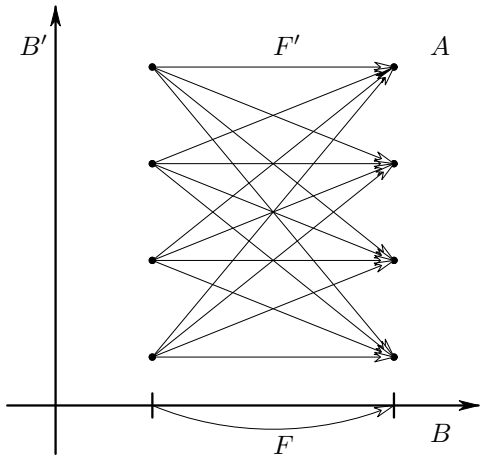
A definíciót úgy is megfogalmazhatjuk, hogy a feladat kiterjesztése az összes olyan $A \times A$-beli pontot tartalmazza, amelynek B-re vett projekciója benne van F-ben.

4.2. A program kiterjesztése

A program kiterjesztésének definíciójában 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.

4.2. DEFINÍCIÓ: PROGRAM KITERJESZTÉSE

Legyen a B állapottér altere az A állapottérnek, és jelölje B' a B kiegészítő

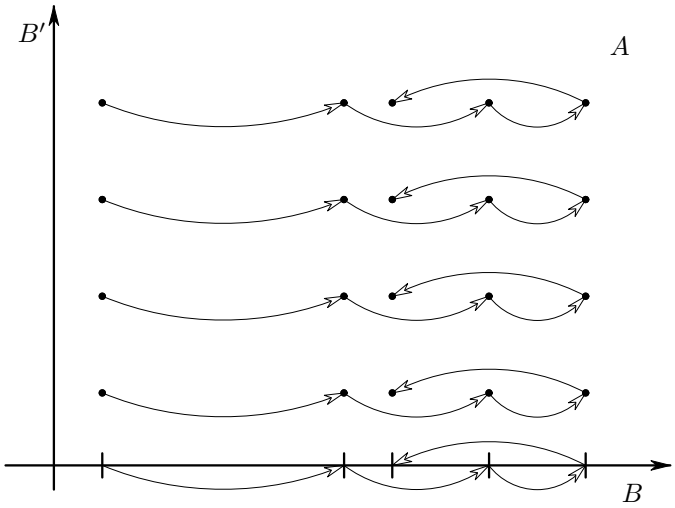


4.1. ábra. Feladat kiterjesztése

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-e a program-tulajdonságot? Erre a kérdésre válaszol az alábbi állítás.

4.1. állítás: *Legyen a B állapottér altere az A állapottérnek, és jelölje B' a B kiegészítő alterét A-ra. Legyen továbbá S program a B állapottéren, és S' az S kiterjesztése A-ra. Ekkor S' program.*

A bizonyítás rendkívül egyszerű, a feladatok között szerepel.

Két azonos állapottéren definiált programot ekvivalensnek neveztünk, ha a programfüggvényük megegyezett. Most ezt a fogalmat is általánosítjuk.

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, ha

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

Nyilvánvaló, hogy a definíció valóban általánosítása az egyszerű esetnek. 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 elhanyagolható – különbség lehet például a hatékonyságukban. Egyáltalán nem mindegy, hogy egy program mennyi ideig fut, és mekkora memóriára van szüksége. A program e jellemzőinek vizsgálatával azonban itt nem foglalkozunk.

A definíciókból közvetlenül adódik a következő állítás:

4.2. 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.

Legyen A egy állapottér, amelynek B altere. A B-n definiált feladatok és programok megfelelőinek tekinthetjük A-n a feladatok és programok kiterjesztéseit A-ra, az A-n definiált feladat megfelelőjének B-n pedig a feladat vetületét B-re. Az A-n definiált programok esetében a B-re való vetítés közvetlenül nem alkalmazható, mivel egy program vetülete nem biztos, hogy program, ugyanis nem biztos, hogy a sorozatok redukáltak. Természetesen, ha $\hat{S} \subseteq A \times A^{**}$ program, akkor az

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

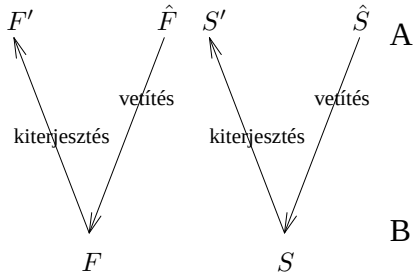
már program, és a B állapottéren S és $\hat{S}$ ekvivalens. Tehát egy $\hat{S} \subseteq A \times A^{**}$ programhoz mindig található olyan $S \subseteq B \times B^{**}$ program, amely vele ekvivalens B-n.

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

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

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

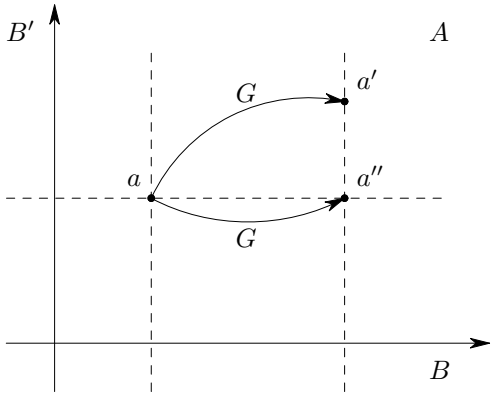
Megvizsgáljuk, milyen esetekben következtethetünk az A állapottéren fennálló megoldásból, ugyanerre a B állapottéren, és fordítva. Ahhoz, hogy a feltételeket megfogalmazhassuk, szükségünk lesz néhány definícióra.



4.3. ábra. Kapcsolat A és B között.

4.4. DEFINÍCIÓ: BŐVÍTETT IDENTITÁS

Legyen B altere A-nak, B' a B kiegészítő altere A-ra, $G \subseteq A \times A$ reláció. 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 definíciók alapján, hogy egy feladat kiterjesztése és egy program kiterjesztésének a programfüggvénye egyaránt bővített identitás.

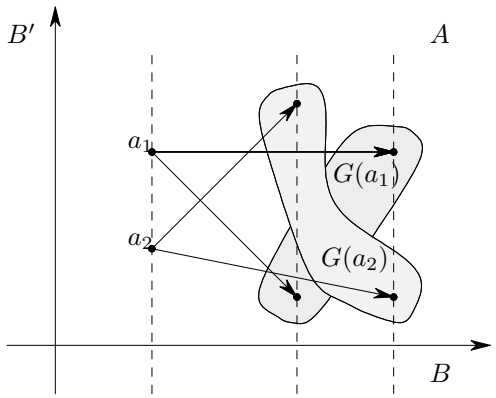
4.5. DEFINÍCIÓ: VETÍTÉSTARTÁS

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

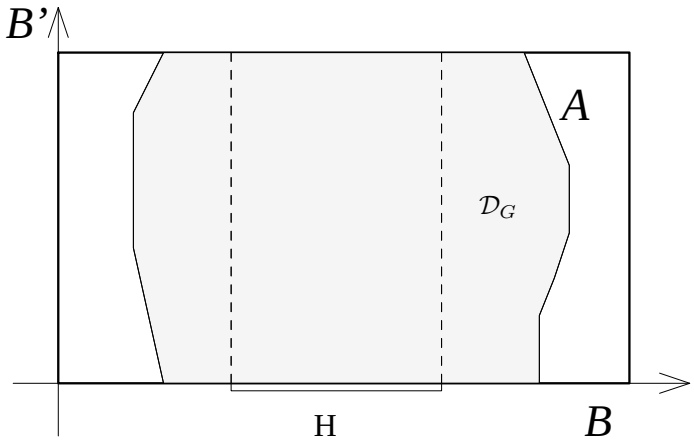
A vetítéstartás nem jelenti azt, hogy a reláció nem függ a kiegészítő altér komponenseitől, hiszen mint a 4.5. ábra mutatja, két azonos vetületű pont képe lehet különböző, csak a vetületük azonos. Ebben az esetben is igaz, hogy egy feladat kiterjesztése vetítéstartó (még a képek is megegyeznek), és a program kiterjesztése, 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 $pr_B^{-1}(H) \subseteq \mathcal{D}_G$.



4.5. ábra. Vetítéstartás



4.6. ábra. Félkiterjesztés

A félkiterjesztés szemléletes jelentése, hogy a kiegészítő 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.

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 a 4.3. ábrának megfelelőek.

4.1. 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 $pr_B(\hat{F}) = F$, $\hat{S} \subseteq A \times A^{**}$ 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 $a \in pr_B^{-1}(b) \cap \mathcal{D}_{\hat{F}}$ 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

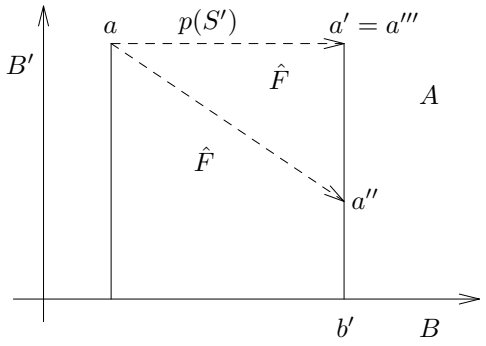
$$\begin{aligned}
&\bigcup_{a\in pr_B^{-1}(b)} p(\hat{S})(a) \subseteq \bigcup_{a\in pr_B^{-1}(b)} \hat{F}(a)\\
\Rightarrow\quad & pr_B\Big(\bigcup_{pr_B^{-1}(b)} p(\hat{S})(a)\Big) \subseteq pr_B\Big(\bigcup_{pr_B^{-1}(b)} \hat{F}(a)\Big)\\
\Rightarrow\quad & \bigcup_{pr_B^{-1}(b)} pr_B(p(\hat{S})(a))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

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.

   A 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 megoldja F-et, adódik, hogy $b' \in F(pr_B(a))$. Ekkor – mivel $\hat{F}$ vetítéstartó B felett, és F az $\hat{F}$ 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 kiterjesztésének 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 megoldja 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 a megoldás második feltétele is teljesül.

Ezzel a (7) állítást is bebizonyítottuk. □

4.4. 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:

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

Mi lesz ennek a 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.

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

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

4.5. A paramétertér kiterjesztése

Természetesen nemcsak az állapottér, de a paramétertér is kiterjeszthető. A specifikáció tétele szempontjából két esetet különböztetünk meg.

Legyen B egy paramétertér és B' egy kiterjesztése. Ha egy feladat specifikálására B helyett B'-t használjuk úgy, hogy az elő- és utófeltételek nem függenek a csak a B'-ben szereplő paraméterváltozóktól, akkor a feladat nem változik, hiszen $\forall b \in B : \forall b' \in pr_B^{-1}(b) : (\lceil Q_{b'} \rceil = \lceil Q_b \rceil$ és $\lceil R_{b'} \rceil = \lceil R_b \rceil)$.

Ellenkező esetben a feladat is megváltozhat. Tegyük fel, hogy az új feladatra teljesülnek a következő feltételek:

$$\forall b \in B : \lceil Q_b \rceil = \bigcup_{b' \in pr_B^{-1}(b)} \lceil Q_{b'} \rceil$$

$$\forall b \in B : \forall b' \in pr_B^{-1}(b)(\lceil R_{b'} \rceil \neq \emptyset \text{ és } \lceil R_{b'} \rceil \subseteq \lceil R_b \rceil),$$

ekkor azt mondjuk, hogy az új feladat *finomítása* a réginek.

4.3. állítás: *Ha egy F feladat finomítása egy G feladatnak, akkor F szigorúbb, mint G.*

4.6. Példák

4.1. példa: $B = \{1,2,3\}, A = B \times \{1,2,3\}$. $F \subseteq B \times B$. $F = \{(1,2),(1,3)\}$. 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 $F = \{((l,k),(m,n)) \mid n = (l \wedge k)\}$ 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, hogy 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 tartalmazás 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. $B = \mathbb{N}, A = B \times \mathbb{N}$. $F \subseteq B \times B$. $F = \{(q, r) \mid r = q + 1\}$. Mi az F kiterjesztettje A-ra?

4.2. Igaz-e, hogy ha $S \subseteq A \times A^{**}$ program, B altere A-nak, akkor S $B \times B$-ra történő projekciójának kiterjesztése A-ra 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? Adja 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 megoldás fogalmának általánosítása

Ebben a fejezetben a megoldás fogalmát általánosítjuk. Eredetileg föltettük, hogy a program és a feladat állapottere azonos. Ezt a kikötést fogjuk két szempontból is gyengíteni.

5.1. A megoldás fogalmának kiterjesztése

Először a kiterjesztési tételek alapján általánosítjuk a megoldás fogalmát. Az állapotterek azonossága helyett csak azt követeljük meg, hogy egy közös állapottérre kiterjesztve a feladatot és a programot, teljesüljenek a megoldás feltételei.

5.1. DEFINÍCIÓ: A MEGOLDÁS FOGALMÁNAK KITERJESZTÉSE

$A = \overset{\times}{_{i \in I}} A_i$ 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, amelynek A és B is altere, és S kiterjesztése C-re – eredeti értelemben – megoldása F C-re való kiterjesztettjének, akkor azt mondjuk, S – *kiterjesztett* értelemben – *megoldása* F-nek.

Ez a definíció az eredeti definíció általánosítása, hiszen $A = B = C$ esetén visszakapjuk azt.

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

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

A kiterjesztett megoldás fogalmának 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 állapotterén is teljesülnek a megoldás feltételei. Ugyanez fordítva már csak bizonyos feltételek teljesülése esetén igaz (5., 6. tétel).

5.2. Megoldás ekvivalens állapottéren

Az ekvivalens direktszorzat definíciójának megfelelően megmondjuk, hogy két állapotteret mikor tekintünk ekvivalensnek.

5.2. DEFINÍCIÓ: EKVIVALENS ÁLLAPOTTÉR

Azt mondjuk, hogy az $A = \overset{\times}{_{i \in I}} A_i$ állapottér *ekvivalens* a $B = \overset{\times}{_{j \in J}} B_j$ állapottérrel, ha létezik olyan $f : I \to J$ kölcsönösen egyértelmű megfeleltetés (bijekció), hogy $\forall i \in I : A_i = B_{f(i)}$.

Azt, hogy A, B, f-re a fenti definíció teljesül, $A \overset{f}{\sim} B$-vel jelöljük.

Ha két állapottér ekvivalens, akkor nemcsak az értelmezési tartományok, hanem az állapotterek között is létezik kölcsönösen egyértelmű megfeleltetés:

$$\gamma_f : B \to A \text{ és } \forall b \in B : \forall i \in I : \gamma_f(b)(i) = b(f(i)).$$

5.3. DEFINÍCIÓ: MEGOLDÁS ÁTNEVEZÉSSEL

Legyenek $A \overset{f}{\sim} B$, $F \subseteq A \times A$ feladat és $S \subseteq B \times B^{**}$ program. Azt mondjuk, hogy S az *f átnevezéssel megoldása* F-nek, ha

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

A definícióban úgy is fogalmazhattunk volna, hogy $\gamma_f \circ p(S) \circ \gamma_f^{(-1)}$ megoldása – az eredeti értelemben – F-nek. A $\gamma_f \circ p(S) \circ \gamma_f^{(-1)}$ reláció jelentését úgy is megfogalmazhatjuk, hogy egy A-beli elemet „átnevezünk” B-belivé, alkalmazzuk rá a programfüggvényt, s a kapott elemeket „visszanevezzük” A-belivé.

Világos, hogy a definíció az eredeti általánosítása, hiszen ha $A = B$ és $f = id_A$, akkor visszakapjuk az eredetit.

A kiterjesztés és az átnevezés segítségével megfogalmazhatjuk a megoldás általános definícióját:

5.4. DEFINÍCIÓ: AZ ÁLTALÁNOSÍTOTT MEGOLDÁS

Legyen $F \subseteq A \times A$ feladat és $S \subseteq B \times B^{**}$ program. Ha létezik olyan C és D állapottér, hogy $C \overset{f}{\sim} D$, A altere C-nek, B altere D-nek, és S kiterjesztése C-re átnevezéssel megoldása F C-re való kiterjesztettjének, akkor azt mondjuk, S – *általános* értelemben – *megoldása* F-nek.

5.3. Reláció szerinti megoldás

Nem csak átnevezéssel és nem csak ekvivalens állapottereket feleltethetünk meg egymásnak. A megoldást definiálhatjuk tetszőleges állapottereken definiált program és feladat esetén is, ha az állapottereket valahogy megfeleltetjük egymásnak.

5.5. DEFINÍCIÓ: RELÁCIÓ SZERINTI MEGOLDÁS

Legyen A és B tetszőleges állapottér, $F \subseteq A \times A$, $S \subseteq B \times B^{**}$ program és $\gamma \subseteq B \times A$. Azt mondjuk, hogy S *γ szerint megoldása* F-nek, ha

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

Emlékeztetünk arra, hogy a kompozíció és a szigorú kompozíció megegyezik, ha legalább az egyik reláció függvény. Ezért a megoldás átnevezéssel a reláció szerinti megoldás speciális esete. Így a következő tétel arra is alkalmazható.

5.1. TÉTEL: RELÁCIÓ SZERINTI MEGOLDÁS TÉTELE

Legyen F tetszőleges feladat, az állapottere A, egy paramétertere B, elő- és utófeltétele pedig Q_b és R_b. Legyen $S \subseteq C \times C^{**}$ program és $\gamma \subseteq C \times A$ tetszőleges olyan reláció, amelyre $\mathcal{D}_F \subseteq \mathcal{R}_\gamma$. Definiáljuk a következő függvényeket:

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

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

Bizonyítás: A γ szerinti megoldás definíciója két pontjának teljesülését kell belátnunk.

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

   Legyen $a \in \mathcal{D}_F$. Ekkor $\exists b \in B : a \in \lceil Q_b \rceil$. Mivel $\mathcal{D}_F \subseteq \mathcal{R}_\gamma$,

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

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

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

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

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

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

□

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

Legyen $T = \{1, 2\}$, F állapottere $A = T$, és a paramétertér is legyen ugyanez: $B = T$. Legyen F specifikációja:

$$\begin{aligned} \lceil Q_1 \rceil &= \{1\}, & \lceil R_1 \rceil &= \{2\}; \\ \lceil Q_2 \rceil &= \emptyset, & \lceil R_2 \rceil &= \{1\}. \end{aligned}$$

Ekkor $\mathcal{D}_F = \{1\}$ és $F(1) = \{2\}$. Legyen $E = \{a, b\}$, $C = E$ és

$$\gamma(a) = \gamma(b) = \{1, 2\}.$$

Tegyük fel, hogy $S \subseteq C \times C^{**}$, és rendelje S az állapottere minden pontjához az önmagából álló egy hosszúságú sorozatot. Ekkor

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

tehát

$$\begin{aligned} Q_1 \circ \gamma &\Rightarrow lf(S, R_1 \circ \gamma), \\ Q_2 \circ \gamma &\Rightarrow lf(S, R_2 \circ \gamma). \end{aligned}$$

Az viszont könnyen látható, hogy

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

tehát S nem oldja meg F-et γ szerint.

6. fejezet

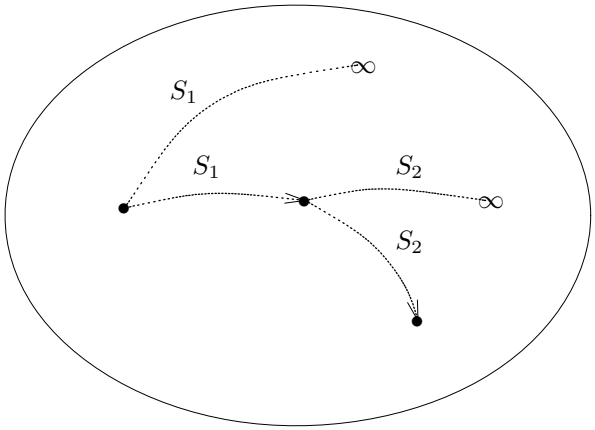
Programkonstrukciók

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

6.1. Megengedett konstrukciók

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

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



6.1. ábra. Szekvencia

6.1. 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 $\forall a \in A$:

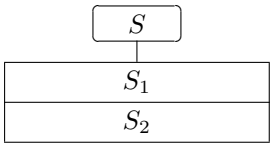
$$\begin{aligned} S(a) &= \{\alpha \in A^{\infty} \mid \alpha \in S_1(a)\} \ \cup \\ &\cup \ \{\chi_2(\alpha, \beta) \in A^{**} \mid \alpha \in S_1(a) \cap A^{*} \text{ és } \beta \in S_2(\tau(\alpha))\}. \end{aligned}$$

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:



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

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

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

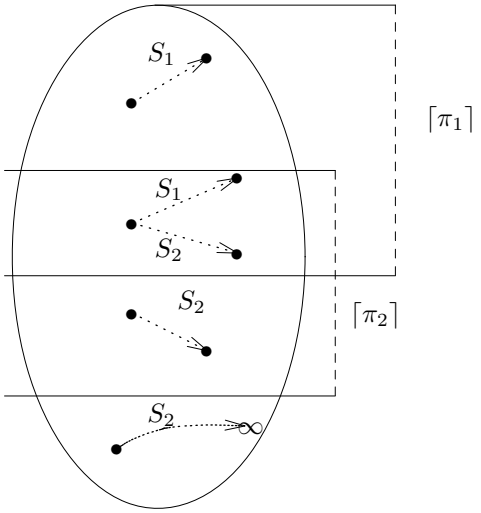
és

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

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 hozzárendel, amelynek a feltételrésze ebben a pontban igaz. Ezért ha a feltételek nem diszjunktak, akkor a determinisztikus programokból képzett elágazás lehet nemdeterminisztikus is.

Az elágazás definícióját így is felírhattuk volna:

$$IF = \bigcup_{i=1}^{n} S_i|_{\lceil \pi_i \rceil} \cup \{\langle a, \ldots \rangle \in A^{**} \mid a \in \bigcap_{i=1}^{n} \lceil \neg\pi_i \rceil\}.$$

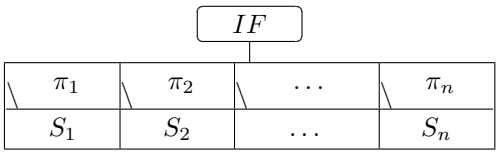


6.2. ábra. Elágazás

Ha csak olyan programokból képzünk elágazást, amelyek csak véges sorozatokat rendelnek az állapottér minden pontjához, az elágazás akkor is rendelhet valamelyik ponthoz (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 gyakorlatbab 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 $\alpha^i \in A^*$ $(i \in \mathbb{N})$,

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

6.3. DEFINÍCIÓ: CIKLUS

Legyen $\pi : A \to \mathbb{L}$ 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

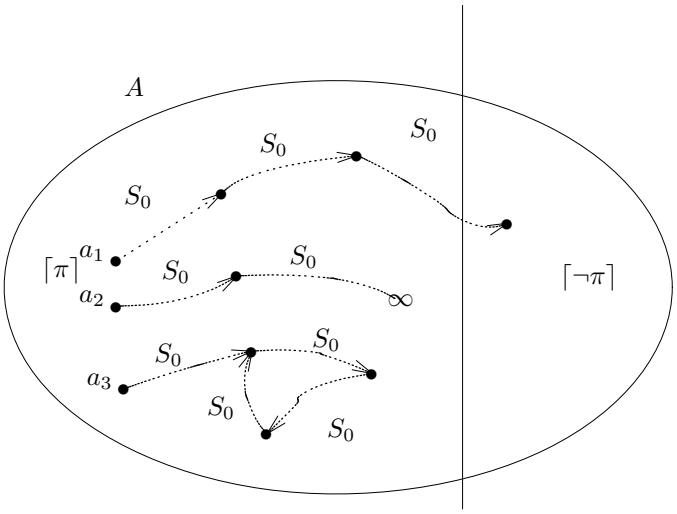
- $\forall a \notin \lceil\pi\rceil$:

$$DO(a) = \{\langle a \rangle\}$$

- $\forall a \in \lceil\pi\rceil$:

$$\begin{aligned} DO(a) \;=\; & \{\alpha \in A^{**} \mid \exists \alpha^1, \ldots, \alpha^n \in A^{**} : \alpha = \chi_n(\alpha^1, \ldots, \alpha^n) \text{ és} \\ & \alpha^1 \in S_0(a) \text{ és } \forall i \in [1..n-1] : \ (\alpha^i \in A^* \text{ és} \\ & \alpha^{i+1} \in S_0(\tau(\alpha^i)) \text{ és } \tau(\alpha^i) \in \lceil\pi\rceil) \text{ és } (\alpha^n \in A^\infty \text{ vagy} \\ & (\alpha^n \in A^* \text{ és } \tau(\alpha^n) \notin \lceil\pi\rceil))\} \ \cup \\ \cup \;\; & \{\alpha \in A^\infty \mid \forall i \in \mathbb{N} : \exists \alpha^i \in A^* : \alpha = \chi_\infty(\alpha^1, \alpha^2, \ldots) \text{ és} \\ & \alpha^1 \in S_0(a) \ \text{ és } \forall i \in \mathbb{N} : \ (\alpha^i \in A^* \text{ és } \alpha^{i+1} \in S_0(\tau(\alpha^i)) \text{ és} \\ & \tau(\alpha^i) \in \lceil\pi\rceil)\}. \end{aligned}$$

Első ránézésre a definíció kissé bonyolult. Nézzük meg alaposabban! A ciklus-



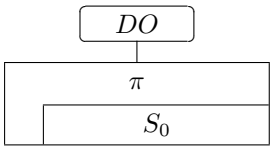
6.3. ábra. Ciklus

magot – S_0 – véges sokszor alkalmazhatjuk egymás után, mert vagy olyan pontba jutunk, ahol a ciklusfeltétel – π – nem teljesül (a 6.3. ábrán az a_1 pont), vagy az S_0 végtelen sorozatot rendel egy ponthoz (a 6.3. ábrán az a_2 pont). Ezeket az eseteket tartalmazza a definícióban az első halmaz. A harmadik lehetőség (a 6.3. ábrán az a_3 pont) az, hogy a fentiek egyike sem teljesül, azaz az S_0 végtelen sokszor alkalmazható egymás után. Ezt az esetet tartalmazza a definícióban a második halmaz.

Ezek alapján azt is megállapíthatjuk, 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 (lásd harmadik lehetőség), azaz kétféle oka is lehet a „végtelen ciklusnak”.

Megjegyezzük még, hogy a különböző programnyelvekben többféle ciklus is létezik; amit most definiáltunk, az az úgynevezett „while típusú” ciklus.

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 előzőekben definiált három konstrukciós művelet meglevő programokból valóban programokat hoz létre.

6.1. 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

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 része $A \times A^{**}$-nak. A továbbiakban a három kritérium teljesülését vizsgáljuk meg mindhárom konstrukció esetén.

1. Legyen $a \in A$ tetszőleges, ekkor $S_1(a)$-nak van legalább egy eleme. Ha ez a sorozat végtelen, a definíció első halmaza nem üres, ha véges, létezik végpontja, amihez S_2 hozzárendel legalább egy sorozatot, és ezért a definíció második halmaza nem üres, tehát $S(a)$ nem üres.

   Legyen $\alpha \in S(a)$. Ekkor két eset lehetséges:

   - Ha $\alpha \in S_1(a)$, 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, ekkor vagy valamelyik π_i feltétel igaz a-ra,és így $w_i(a)$ nem üres, vagy $w_0(a)$ nem üres, tehát $IF(a)$ nem üres.

   Legyen $\alpha \in IF(a)$. 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, ekkor vagy minden $n \in \mathbb{N}$-re teljesül a definíció második halmazának a feltétele, vagy ha nem, akker teljesül az elsőé, azaz a két halmaz közül legalább az egyik nem üres, tehát $DO(a)$ nem üres.

   Legyen $\alpha \in DO(a)$.

   - Tegyük fel, hogy $a \notin \lceil \pi \rceil$. Ekkor $\alpha = \langle a \rangle$, így az előírt tulajdonságok triviálisan teljesülnek.
   - Tegyük fel, hogy $a \in \lceil \pi \rceil$. 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) $\exists n \in \mathbb{N} : \alpha = \chi_n(\alpha^1, \ldots, \alpha^n)$ és $\forall i \in [1..n-1] : \alpha^i \in A^*$ és $\alpha_n \in A^\infty$: 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.

□

6.2. A programkonstrukciók programfüggvénye

Miután beláttuk, hogy meglevő programokból a programkonstrukciók segítségével új programokat készíthetünk, vizsgáljuk meg, 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.

6.2. 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

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

tehát:

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

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

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

□

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 $A = \{1, 2\}$,

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

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, 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 programfüggvényét adja meg a következő tétel.

6.3. 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

- $\mathcal{D}_{p(IF)} = \{a \in A \mid a \in \bigcup_{i=1}^{n} \lceil \pi_i \rceil$ és $\forall i \in [1..n] : \ a \in \lceil \pi_i \rceil \Rightarrow a \in \mathcal{D}_{p(S_i)}\}$.
- $\forall a \in \mathcal{D}_{p(IF)}$:

$$p(IF)(a) = \bigcup_{i=1}^{n} p(S_i)|_{\lceil \pi_i \rceil}(a).$$

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

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

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

$$p(IF)(a) = \bigcup_{i=1}^{n} \{\tau(\alpha) \mid \alpha \in w_i(a)\} = \bigcup_{i=1}^{n} p(S_i)|_{\lceil \pi_i \rceil}(a).$$

□

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

A ciklus programfüggvényét a feltételre vonatkozó lezárt segítségével fejezzük ki.

6.4. 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

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

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

- Ha $a \notin \lceil \pi \rceil$, akkor
$$p(DO)(a) = \{a\} = \overline{p(S)|\pi}(a).$$
- Ha $a \in \lceil \pi \rceil$, akkor $p(DO)(a) =$

$$\begin{aligned} &= \{\tau(\alpha) \mid \alpha \in A^* \text{ és } \exists n \in \mathbb{N} : \alpha = \chi_n(\alpha^1, \ldots, \alpha^n) \text{ és } \alpha^1 \in S(a) \text{ és} \\ &\qquad \forall i \in [1..n-1] : \alpha^{i+1} \in S(\tau(\alpha^i)) \text{ és } \tau(\alpha^i) \in \lceil \pi \rceil \text{ és } \tau(\alpha^n) \notin \lceil \pi \rceil\} \\ &= \{\tau(\beta) \mid \beta \in A^* \text{ és } \beta_1 = a \text{ és } \forall i \in [1..|\beta| - 1] : \beta_{i+1} \in p(S)(\beta_i) \text{ és} \\ &\qquad \beta_i \in \lceil \pi \rceil \text{ és } \tau(\beta) \notin \lceil \pi \rceil\} = \\ &= \{b \in A \mid \exists k \in \mathbb{N} : b \in (p(S)|\pi)^k(a) \text{ és } b \notin \lceil \pi \rceil\} \\ &= \overline{p(S)|\pi}(a). \end{aligned}$$

□

6.3. Feladatok

6.1. $A = \{1, 2, 3, 4, 5, 6\}$. $\lceil \pi_1 \rceil = \{1, 2, 3, 4\}$. $\lceil \pi_2 \rceil = \{1, 3, 4, 5\}$.

$$\begin{aligned} S_1 = \{ &\quad 1{\to}14 \quad 1{\to}12\ldots \quad 2{\to}2132 \quad 3{\to}36 \\ &\quad 4{\to}463 \quad 4{\to}451 \quad 5{\to}563 \quad 6{\to}612 \quad \}, \\ S_2 = \{ &\quad 1{\to}134 \quad 1{\to}121 \quad 2{\to}2132\ldots \quad 3{\to}36 \\ &\quad 4{\to}463 \quad 4{\to}451\ldots \quad 5{\to}5632 \quad 6{\to}61\ldots \quad \}. \end{aligned}$$

Adja meg az $(S_1; S_2)$, $IF(\pi_1 : S_1, \pi_2 : S_2)$, $DO(\pi_1, S_1)$ programokat és a programfüggvényeiket!

6.2. Legyen $A = \{1, 2, 3, 4, 5, 6\}$, $\lceil \pi_1 \rceil = \{1, 2, 3, 4\}$, $\lceil \pi_2 \rceil = \{2, 3, 4\}$, $\lceil \pi_3 \rceil = \{1, 4, 6\}$ és

$$\begin{aligned} S_1 = \{ &\quad 1{\to}12\ldots \quad 2{\to}23 \quad 3{\to}3456 \\ &\quad 4{\to}463 \quad 5{\to}53 \quad 6{\to}62 \quad \}, \\ S_2 = \{ &\quad 1{\to}12 \quad 2{\to}24 \quad 3{\to}3\ldots \\ &\quad 4{\to}43 \quad 5{\to}5 \quad 6{\to}61 \quad \}, \\ S_3 = \{ &\quad 1{\to}12 \quad 2{\to}2\ldots \quad 3{\to}31 \\ &\quad 4{\to}432 \quad 5{\to}5\ldots \quad 6{\to}63\ldots \quad \}. \end{aligned}$$

Mi lesz $IF(\pi_1 : S_1, \pi_2 : S_2, \pi_3 : S_3)$, $\mathcal{D}_{p(IF)}$, $p(IF)$?

6.3. 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?

6.4. $S = (S_1; S_2)$. Igaz-e, hogy

a) $\mathcal{D}_{p(S)} = \lceil lf(S_1, \mathcal{P}(\mathcal{D}_{p(S_2)}))\rceil$?

b) tetszőleges R utófeltételre: $lf((S_1; S_2), R) = lf(S_1, lf(S_2, R))$?

6.5. Van-e olyan program, amely felírható szekvenciaként is, elágazásként is, és felírható ciklusként is?

6.6. Igaz-e, hogy minden program felírható szekvenciaként is, elágazásként is, és felírható ciklusként is?

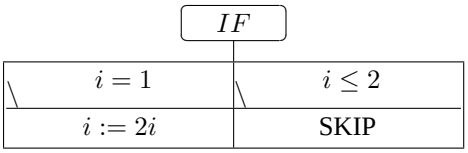
6.7. $IF = (\pi_1 : S_1, \ldots, \pi_n : S_n)$. Igaz-e, hogy $\mathcal{D}_{p(IF)} = \bigcup_{k=1}^{n} (\lceil \pi_k \rceil \cap \mathcal{D}_{p(S_k)})$?

6.8. 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 $\mathcal{D}_{p(S)} = \emptyset$!

6.9. $IF = (\pi_1 : S_1, \ldots, \pi_n : S_n)$. $S = S_1 \cup S_2 \cup \cdots \cup S_n$. Igaz-e, hogy $p(IF)$ része $p(S)$-nek?

6.10. Igaz-e? Ha $IF = (\pi_1 : S_1, \pi_2 : S_2)$, akkor $\mathcal{D}_{p(IF)} = (\lceil \pi_1 \rceil \cap \lceil \pi_2 \rceil \cap \mathcal{D}_{p(S_1)} \cap \mathcal{D}_{p(S_2)}) \cup (\mathcal{D}_{p(S_1)} \cap (\lceil \pi_1 \rceil \setminus \lceil \pi_2 \rceil)) \cup (\mathcal{D}_{p(S_2)} \cap (\lceil \pi_2 \rceil \setminus \lceil \pi_1 \rceil))$?

6.11. $A = \{1, 2, 3, 4\}$.



Milyen sorozatokat rendel S_1, S_2, IF az állapottér egyes pontjaihoz?

6.12. $S = (S_1; S_2)$. S_1 megoldja F_1-et, és S_2 megoldja F_2-t. Megoldja-e S az

a) $F = F_2 \circ F_1$ feladatot?

b) $F = F_2 \odot F_1$ feladatot?

6.13. $S = (S_1; S_2)$. S megoldása az $(F_2 \odot F_1)$ feladatnak.
Megoldja-e S_1 az F_1-et, illetve S_2 az F_2-t?

6.14. $IF = (\pi_1 : S_1, \ldots, \pi_n : S_n)$. $F \subseteq A \times A$ 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 $\mathcal{D}_F \subseteq \lceil \pi_1 \vee \pi_2 \vee \cdots \vee \pi_n \rceil$?

6.15. $IF = (\pi_1 : S_1, \ldots, \pi_n : S_n)$. $F \subseteq A \times A$ feladat. IF megoldja az F feladatot. Igaz-e, hogy $\forall k \in [1..n] :$ S_k megoldja az $F|_{\lceil \pi_k \rceil}$ feladatot?

6.16. $IF = (\pi_1 : S_1, \ldots, \pi_n : S_n)$. $F_1, \ldots, F_k \subseteq A \times A$ 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?

6.17. $IF = (\pi_1 : S_1, \ldots, \pi_n : S_n)$. $F \subseteq A \times A$ 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?

6.18. $IF = (\pi_1 : S_1, \ldots, \pi_n : S_n)$. $F_1, \ldots, F_n \subseteq A \times A$ 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?

6.19. 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?

6.20. 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?

6.21. $F \subseteq A \times A$ 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?

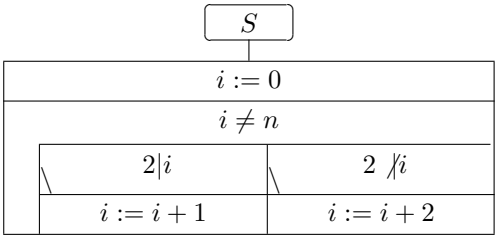
6.22. Legyen $DO = (\pi, S)$! Igaz-e, hogy

a) $p(DO) \subseteq p(S)$?

b) $p(S) \subseteq p(DO)$?

6.23. $A = \underset{i}{\mathbb{N}_0} \times \underset{n}{\mathbb{N}_0}$

$S = ((i := 0; DO(i \neq n, IF(\ 2|i :\ i := i + 1,\ \ 2 \not| i\ :\ \ i := i + 2))))$



Milyen sorozatokat rendel S a $(2, 4)$, illetve a $(3, 7)$ ponthoz?

7. fejezet

Levezetési szabályok

Ebben a fejezetben 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.

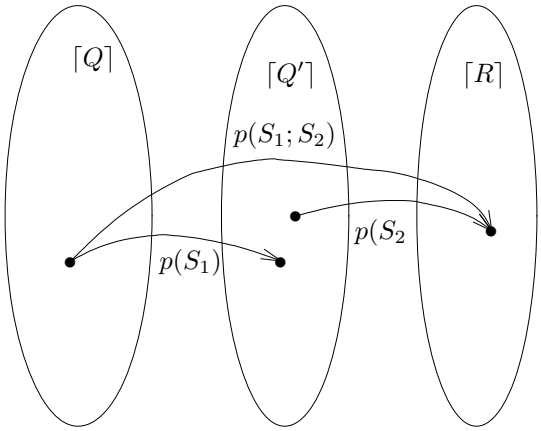
7.1. TÉTEL: A SZEKVENCIA LEVEZETÉSI SZABÁLYA

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



7.1. ábra. A szekvencia levezetési szabálya

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.

7.2. 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, amelyre

(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 módon 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.

7.3. TÉTEL: AZ ELÁGAZÁS LEVEZETÉSI SZABÁLYA
Legyen $IF = (\pi_1 : S_1, \ldots, \pi_n : S_n)$, és adott Q, R állítás A-n. Ha

(1) $Q \Rightarrow \bigvee_{i=1}^{n} \pi_i$,

(2) $\forall i \in [1..n] : \; Q \wedge \pi_i \Rightarrow lf(S_i, R)$,

akkor

$$Q \Rightarrow lf(IF, R).$$

Bizonyítás: Legyen $q \in \lceil Q \rceil$, és tegyük fel, hogy valamelyik feltétel igaz q-ra, azaz $\exists i \in [1..n] : q \in \lceil \pi_i \rceil$. Ekkor $q \in \mathcal{D}_{p(IF)}$, ugyanis

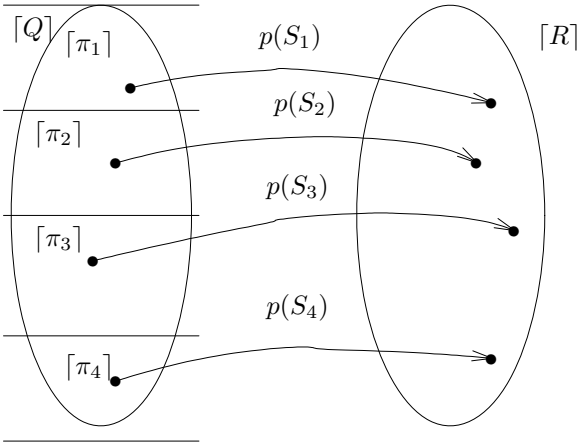
$$\forall j \in [1..n] : q \in \lceil \pi_j \rceil \Rightarrow q \in \lceil lf(S_j, R) \rceil \Rightarrow q \in \mathcal{D}_{p(S_j)}.$$

Másrészt mivel $\forall j \in [1..n] : q \in \lceil \pi_j \rceil \Rightarrow p(S_j)(q) \subseteq \lceil R \rceil$:

$$p(IF)(q) = \bigcup_{\substack{j \in [1..n] \\ q \in \lceil \pi_j \rceil}} p(S_j)(q) \subseteq \lceil R \rceil,$$

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



7.2. ábra. Az elágazás levezetési szabálya

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.

7.4. 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

$$Q \Rightarrow lf(IF, R).$$

Ekkor

(1) $Q \Rightarrow \bigvee_{i=1}^{n} \pi_i$,

(2) $\forall i \in [1..n] : \; Q \wedge \pi_i \Rightarrow lf(S_i, R)$.

Bizonyítás: Ha $q \in \lceil Q \rceil$, akkor $q \in \mathcal{D}_{p(IF)}$, vagyis $\exists i \in [1..n] : q \in \lceil \pi_i \rceil$. Tehát (1) teljesül.

Ezután belátjuk, hogy (2) is teljesül. Tegyük fel indirekt módon, 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.

□

Az előző két konstrukcióhoz hasonlóan most megvizsgáljuk, milyen kapcsolat van a ciklus és a ciklusmag leggyengébb előfeltétele között.

7.5. 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) $P \wedge \neg\pi \Rightarrow R$,

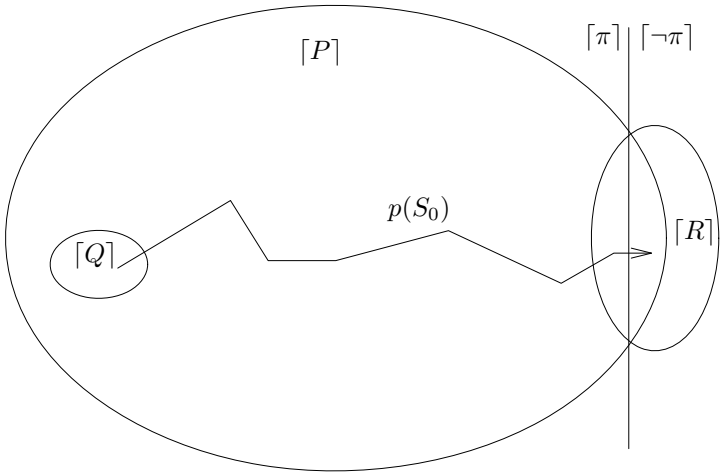
(3) $P \wedge \pi \Rightarrow t > 0$,

(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

$$Q \Rightarrow lf(DO, R).$$



7.3. ábra. A ciklus levezetési szabálya

A tétel szerint azt kellene belátnunk, hogy:

(1) $\forall q \in \lceil Q \rceil : q \in \mathcal{D}_{p(DO)} = \mathcal{D}_{\overline{p(S_0)|_\pi}}$ és

(2) $\forall q \in \lceil Q \rceil : p(DO)(q) \subseteq \lceil R \rceil$, azaz $\overline{p(S_0)|_\pi}(q) \subseteq \lceil R \rceil$.

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 $q \in \lceil P \rceil$. Ekkor

$$\forall k \in \mathbb{N}_0 : g^k(q) \subseteq \lceil P \rceil.$$

Bizonyítás: k szerinti teljes indukcióval:

- $k = 0$: $g^0(q) = \{q\} \subseteq \lceil P \rceil$.
- 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 $g(r) = \emptyset \subseteq \lceil P \rceil$.

– $r \in \lceil P \wedge \pi \rceil$. Ekkor (4) miatt $r \in \mathcal{D}_{p(S_0)}$ és $g(r) = p(S_0)(r) \subseteq \lceil P \rceil$.

Tehát: $g^{k+1}(q) \subseteq \lceil P \rceil$.

□

2. állítás: Legyen $q \in \lceil P \rceil$. Ekkor

$$\exists n \in \mathbb{N} : g^n(q) = \emptyset.$$

Bizonyítás: Indirekt: tegyük fel, hogy $\forall n \in \mathbb{N}_0 : g^n(q) \neq \emptyset$. Ekkor $\forall n \in \mathbb{N}_0 : \exists :$

$$\max_{b \in g^n(q)} t(b).$$

Ez $n = 0$ esetén nyilván igaz. Ha

$$m = \max_{b \in g^n(q)} t(b),$$

akkor, mivel $g(g^n(q)) \subseteq p(S_0)(g^n(q))$, az (5) alapján $\forall x \in g^{n+1}(q) : t(x) < m$, tehát a maximum minden n-re létezne, sőt szigorúan monoton csökkenne.

Ez viszont ellentmond (3)-nak, hiszen t egész értékű függvény. □

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 (4) miatt $q \in \lceil lf(S_0, P) \rceil$, azaz $q \in \mathcal{D}_{p(S_0)}$ és így $p(S_0)|_\pi(q) = g(q)$. Tehát a 2. állítás alapján

  $$\exists n \in \mathbb{N}_0 : (p(S_0)|_\pi)^n(q) = \emptyset,$$

  azaz

  $$q \in \mathcal{D}_{p(S_0)|_\pi} = \mathcal{D}_{p(DO)}.$$

  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, amelyet 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.

Azonban ha egy ciklus programfüggvénye megegyezik a ciklusmag ciklusfeltételre vonatkozó korlátos lezártjával, akkor a ciklus levezethető.

7.6. 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.

Megjegyzés: mivel a ciklus levezetési szabálya nem megfordítható, nem minden ciklus vezethető le, ez azonban nem jelent érdemi korlátozást. Ha van egy ciklus, amely megoldása egy feladatnak, akkor biztosan van olyan ciklus is, amelynek a feltétele ugyanaz, szintén megoldása a feladatnak, és a programfüggvénye korlátos lezárt.

A ciklus definíciójából látszik, ha $S_1 \subseteq S_2$, akkor $DO_1(\pi, S_1) \subseteq DO_2(\pi, S_2)$, és a 2.7. példa szerint DO_1 megoldása minden olyan feladatnak, aminek DO_2 megoldása. Tehát S_1 akár determinisztikus is lehet, és akkor a lezártja $p(S_1)_{|\pi}$-nek biztosan korlátos.

7.1. Feladatok

7.1. Tegyük fel, hogy teljesül a ciklus levezetési szabályának mind az öt feltétele, és Q igazsághalmaza nem üres. Lehet-e üres a

a) $\lceil P \wedge R \rceil$ halmaz?

b) $\lceil P \wedge \neg\pi \wedge R \rceil$ halmaz?

7.2. Tegyük fel, hogy teljesül a ciklus levezetési szabályának mind az öt feltétele, és $(Q \wedge \pi)$ igazsághalmaza nem üres. Lehet-e üres az $lf(S_0, P)$ és $lf(DO, R)$ igazsághalmazának metszete?

7.3. Tegyük fel, hogy teljesül a ciklus levezetési szabályának mind az öt feltétele. Legyen $g = p(S_0)|_{\lceil \pi \rceil}$ 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) $\exists k \in \mathbb{N}_0 : \; k \leq t(q)$ és $g^k(q) \subseteq \lceil \neg\pi \rceil$?

7.4. 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 $\lceil Q \rceil \cap \lceil R \rceil = \emptyset$ és $\lceil Q \rceil \cap \lceil Q' \rceil \neq \emptyset$ és $\lceil Q' \rceil \cap \lceil R \rceil \neq \emptyset$?
Indokolja, ha nem, és írjon rá példát, ha igen!

7.5. $A = \underset{x}{\mathbb{Z}} \times \underset{y}{\mathbb{N}_0}$

$B = \underset{x'}{\mathbb{Z}} \times \underset{y'}{\mathbb{N}_0}$

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

$R: \quad (x = x' - y' \wedge y = 0)$

$S_0 = \quad \{((x,y), <(x,y),(x-1,y),(x-1,y-1)>) \mid x \in \mathbb{Z} \text{ és } y \in \mathbb{N}\} \cup$
$\{((x,0), <(x,0)>) \mid x \in \mathbb{Z}\}$

$DO = \quad \{((x,y), <(x,y),(x-1,y),(x-1,y-1),(x-2,y-1),$
$(x-2,y-2),\ldots,(x-y+1,1),(x-y,1)(x-y,0)>) \mid$
$x \in \mathbb{Z} \text{ és } y \in N_0\}.$

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, adjon meg egy megfelelő π-t, P-t és t-t!

7.6. $A = \underset{k}{\mathbb{Z}} \times \underset{x}{\mathbb{Z}} \times \underset{i}{\mathbb{Z}} \times \underset{a}{\mathbb{Z}} \times \underset{b}{\mathbb{Z}}$

$B = \underset{a'}{\mathbb{Z}} \times \underset{b'}{\mathbb{Z}}$

$S = (k := 5; (IF(a > b : x := a - b, a \leq b : x := b - a); i := i + 1))$

$Q : (a = a' \wedge b = b' \wedge i \in [0..1] \wedge |a - b| > 10)$

$R : (a = a' \wedge b = b' \wedge k \cdot i \leq x)$

Bizonyítsuk be, hogy $Q \Rightarrow lf(S, R)$!

8. fejezet

Elemi programok

A 2. fejezetben a programozási feladat kapcsán megfogalmaztuk, hogy mindig feltesszük, létezik a programoknak egy adott halmaza, amelyekből összerakhatjuk a programjainkat (2.7.). Ezeket a programokat szoktuk *primitív programoknak* nevezni.

Ebben a fejezetben bevezetünk néhány elméleti szempontból „egyszerű” programot. Ezeket a programokat a következő fejezetekben gyakorta fogjuk használni, ezért megvizsgáljuk a programfüggvényüket és egy adott utófeltételhez tartozó leggyengébb előfeltételüket.

8.1. Elemi programok

8.1. DEFINÍCIÓ: ELEMI PROGRAM

Eleminek nevezünk egy A állapottéren egy S programot, ha

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

A definíció szerint minden programhoz található vele ekvivalens elemi program, csak a sorozatok közbülső elemeit el kell hagyni, s így lényegében, egy adott szinten minden program elemi.

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

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

8.2. 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.

8.3. 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 valóban nagyon egyszerű, de – talán meglepő módon – mégis jelentős szerepük van. A harmadik speciális elemi program az értékadás. Az előzőeknél bonyolultabb, de sokkal fontosabb is.

Az értékadás definíciójához bevezetünk egy jelölést. Legyen $A = A_1 \times \cdots \times A_n$ és $\forall i \in [1..n] : F_i \subseteq A \times A_i$. Jelöljük $F = (F_1, \ldots, F_n)$-nel a következő $F \subseteq A \times A$ relációt:

$$\mathcal{D}_F = \bigcap_{i=1}^{n} \mathcal{D}_{F_i} \quad \text{és} \quad \forall a \in \mathcal{D}_F : F(a) = \bigtimes_{i \in [1..n]} F_i(a).$$

8.4. DEFINÍCIÓ: ÉRTÉKADÁS

Legyen $A = A_1 \times \cdots \times A_n$ és $F = (F_1, \ldots, F_n)$. Az S program *általános értékadás*, ha $\forall a \in A$:

$$S(a) = \begin{cases} \{red(< a, b >) \mid b \in F(a)\}, & \text{ha } a \in \mathcal{D}_F; \\ \{< a, a, a, \cdots >\}, & \text{ha } a \notin \mathcal{D}_F. \end{cases}$$

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

Az általános értékadáson belül speciális eseteket különböztetünk meg:

- Ha $\mathcal{D}_F = A$, akkor az S programot *értékkiválasztás*nak nevezzük.
- Ha az F reláció függvény, akkor az S programot *értékadás*nak nevezzük.
- 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*.

Az értékadást és a parciális értékadást $a := F(a)$-val, az értékkiválasztást és a parciális értékkiválasztás $a :\in F(a)$-val jelöljük.

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!

8.1. állítás:

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

A bizonyítás triviális (a feladatok között szerepel).

8.2. Elemi programok leggyengébb előfeltétele

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 nemdeterminisztikus, illetve a globális és a parciális esetben. Felhasználva a 3. és a 8. állításokat:

$$\lceil lf(a :\in F(a), R)\rceil = \lceil R \circ F\rceil.$$

Ha $F : A \to A$ függvény, akkor $R \circ F$ függvény, és ezért

$$lf(a := F(a), R) = R \circ F.$$

Ha F parciális függvény, tehát az értékadás parciális, akkor ennek leggyengébb előfeltétele:

$$\forall b \in A : lf(a := F(a), R)(b) = \begin{cases} R \circ F(b), & \text{ha } b \in \mathcal{D}_F; \\ hamis, & \text{ha } b \notin \mathcal{D}_F. \end{cases}$$

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

$$lf(a :\in F(a), R) = \begin{cases} igaz, & \text{ha } \forall b \in F(a) : R(a); \\ hamis & \text{egyébként.} \end{cases}$$

Ugyanez parciális esetben:

$$lf(a :\in F(a), R) = \begin{cases} igaz, & \text{ha } a \in \mathcal{D}_F \text{ és } \forall b \in F(a) : R(a); \\ hamis & \text{egyébként.} \end{cases}$$

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

$$A = \underset{x}{\mathbb{Z}} \times \underset{l}{\mathbb{L}}$$

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 a kompozíció képzésének megfelelően: 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

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

Parciális értékadás esetén:

$$lf(x_{i_1}, \ldots, x_{i_m} := F_{i_1}(x_1, \ldots, x_n), \ldots, F_{i_m}(x_1, \ldots, x_n), R) = (x_1, \ldots, x_n) \in \mathcal{D}_F \wedge R^{x_{i_1} \leftarrow F_{i_1}(x_1, \ldots, x_n), \ldots, x_{i_m} \leftarrow F_{i_m}(x_1, \ldots, x_n)}$$

Az értékkiválasztás kicsit bonyolultabb:

$$lf(x_{i_1}, \ldots, x_{i_m} :\in F_{i_1}(x_1, \ldots, x_n), \ldots, F_{i_m}(x_1, \ldots, x_n), R) = \forall (e_{i_1}, \ldots, e_{i_m}) \in \times_{j=1}^{m} F_{i_j}(x_1, \ldots, x_n) : R^{x_{i_1} \leftarrow e_{i_1} \ldots, x_{i_m} \leftarrow e_{i_m}}$$

Végül a parciális érték kiválasztás:

$$lf(x_{i_1}, \ldots, x_{i_m} :\in F_{i_1}(x_1, \ldots, x_n), \ldots, F_{i_m}(x_1, \ldots, x_n), R) = (x_1, \ldots, x_n) \in \mathcal{D}_F \wedge \forall (e_{i_1}, \ldots, e_{i_m}) \in \times_{j=1}^{m} F_{i_j}(x_1, \ldots, x_n) : R^{x_{i_1} \leftarrow e_{i_1} \ldots, x_{i_m} \leftarrow e_{i_m}}$$

A továbbiakban, ha mást nem mondunk, primitív programnak tekintjük azokat az – esetleg parciális – értékadásokat, amelyek jobb oldalán a szokásos aritmetikai, logikai kifejezésekkel felírt függvények állnak.

8.3. Az értékadás mint feladatspecifikáció

Az értékadás lényegében egy programfüggvényével definiált program. Kézenfekvő, hogy ezt a programfüggvényt feladatnak is tekinthetjük. Egy értékadás által definiált feladat könnyen átírható a specifikáció tételének megfelelő formára is.

Legyen $x_1, \ldots, x_n := F_i(x_1, \ldots, x_n), \ldots, F_n(x_1, \ldots, x_n) \subseteq A \times A$ egy értékadás. Az A állapotér változói értelemszerűen $x_1, \ldots, x_n$.

Tekintsük a következő feladatspecifikációt:

$$\begin{array}{lcccl} A = & A_1 & \times \ldots \times & A_n & \\ & x_1 & \ldots & x_n & \\ B = & B_1 & \times \ldots \times & B_n & \\ & x'_1 & \ldots & x'_n & \end{array}$$

$Q : (x_1 = x'_1 \wedge \cdots \wedge x_n = x'_n)$

$R : (x_1 = F_1(x'_1, \ldots x'_n) \wedge \cdots \wedge \times_n = F_n(x'_1, \ldots x'_n))$

Nyilvánvaló, hogy éppen az F függvényt specifikáltuk.

Ha nem értékadásról van szó, hanem értékkiválasztásról, akkor az utófeltételekben nem $=$, hanem $\in$ szerepel.

A parciális esetekben az előfeltételeket célszerű leszűkíteni az F értelmezési tartományára az $x_i \in \mathcal{D}_{F_i}$ feltételek hozzávételével.

8.4. Feladatok

8.1. $A = \{1, 2, 3\}, B = \{a, b\}, C = A \times B$. 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?

8.2. Tekintsük az alábbi állapotteret:

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

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?

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

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

9. fejezet

A programkonstrukciók és a kiszámíthatóság

Ebben a fejezetben 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, és ezzel megmutatjuk azt is, hogy ha egy feladat megoldható, akkor megoldható levezetéssel is.

A kiszámíthatóság fogalmát általában a parciálisan rekurzív függvények bevezetésén keresztül szokás megadni. Más definíciós lehetőségekről, például a Turing-gépről, bizonyítható, hogy a kiszámíthatóság szempontjából egyenértékűek a parciálisan rekurzív függvényekkel.

Felhívjuk a figyelmet arra, hogy ebben a fejezetben függvény alatt mindig parciális függvényt értünk. A mindenütt definiált függvényekre a totális jelzőt használjuk.

9.1. Parciálisan rekurzív függvények

A kiszámíthatóság parciálisan rekurzív függvényekre alapozott 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:

$g \circ f \in \mathbb{N}^m \to \mathbb{N}^k$, $\mathcal{D}_{g\circ f} = \{x \in \mathcal{D}_f \mid f(x) \in \mathcal{D}_g\}$ 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).

Direktszorzat. 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 direktszorzata $(f_1, f_2, \ldots, f_k) \in \mathbb{N}^m \to \mathbb{N}^{n_1} \times \cdots \times \mathbb{N}^{n_k}, \mathcal{D}_{(f_1,f_2,\ldots,f_k)} = \bigcap_{i=1}^{k} \mathcal{D}_{f_i}$ 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

$$\begin{aligned} \varrho(f,g)(x_1,\ldots,x_n,1) &= f(x_1,\ldots,x_n), \\ \varrho(f,g)(x_1,\ldots,x_n,k+1) &= g(x_1,\ldots,x_n,k,\varrho(f,g)(x_1,\ldots,x_n,k)). \end{aligned}$$

$\varrho(f, g)$ értelmezési tartománya azon pontok halmaza, ahonnan 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 segítségével már definiálható a parciális rekurzív függvények halmaza.

9.1. 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 operátoroknak parciális rekurzív függvényekre történő véges sokszori alkalmazásával.

9.2. 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:

$$A = \underbrace{\mathbb{N}}_{x_1} \times \ldots \times \underbrace{\mathbb{N}}_{x_m} \times \underbrace{\mathbb{N}}_{y_1} \times \ldots \times \underbrace{\mathbb{N}}_{y_m}$$

$$B = \underset{x'_1}{\mathbb{N}} \times \ldots \times \underset{x'_m}{\mathbb{N}}$$
$$Q : (x_1 = x'_1 \wedge \cdots \wedge x_m = x'_m \wedge (x_1, \ldots, x_m) \in \mathcal{D}_f)$$
$$R : (Q \wedge (y_1, \ldots, y_n) = f(x'_1, \ldots, x'_m))$$

Feltesszük, hogy az alapfüggvényeket kiszámító programok benne vannak a megengedett programok halmazában. Ezt nyugodtan megtehetjük, hiszen minden programozási nyelv tartalmaz ilyen utasításokat.

Megmutatjuk, hogy az elemi függvény-operátorok (kompozíció, direktszorzat, 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 a $g \circ f$ által meghatározott feladat specifikációja:

$$A = \underset{x_1}{\mathbb{N}} \times \ldots \times \underset{x_m}{\mathbb{N}} \times \underset{y_1}{\mathbb{N}} \times \ldots \times \underset{y_k}{\mathbb{N}}$$
$$B = \underset{x'_1}{\mathbb{N}} \times \ldots \times \underset{x'_m}{\mathbb{N}}$$
$$Q : (x_1 = x'_1 \wedge \cdots \wedge x_m = x'_m \wedge (x_1, \ldots, x_m) \in \mathcal{D}_{g \circ f})$$
$$R : (Q \wedge (y_1, \ldots, y_n) = (g \circ f)(x'_1, \ldots, x'_m))$$

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 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 szabályának 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.

Direktszorzat. 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 direktszorzata által meghatározott feladat specifikációja:

$$A = \underset{x_1}{\mathbb{N}} \times \ldots \times \underset{x_m}{\mathbb{N}} \times \underset{y_{11}}{\mathbb{N}} \times \ldots \times \underset{y_{1n_1}}{\mathbb{N}} \times \ldots \times \underset{y_{k1}}{\mathbb{N}} \times \ldots \times \underset{y_{kn_k}}{\mathbb{N}}$$
$$B = \underset{x'_1}{\mathbb{N}} \times \ldots \times \underset{x'_m}{\mathbb{N}}$$
$$Q : (x_1 = x'_1 \wedge \cdots \wedge x_m = x'_m \wedge (x_1, \ldots, x_m) \in \mathcal{D}_{(f_1, \ldots, f_k)})$$
$$R : (Q \wedge ((y_{11}, \ldots, y_{1n_1}), \ldots, (y_{k1}, \ldots, y_{kn_k})) = (f_1, \ldots, f_k)(x'_1, \ldots, x'_m))$$

Tegyük fel, hogy a komponensfüggvények kiszámíthatók jól konstruált programmal. Jelölje $y_{i1}, \ldots, y_{in_i} := f_i(x_1, \ldots, x_m)$ 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 előfeltétel:

$$Q_k : (Q \wedge ((y_{11}, \ldots, y_{1n_1}), \ldots, (y_{k-1\,1}, \ldots, y_{k-1\,n_{k-1}}), f_k(x_1, \ldots, x_m)) = \\ (f_1, \ldots, f_k)(x'_1, \ldots, x'_m) \wedge (x_1, \ldots, x_m) \in \mathcal{D}_{f_k})$$

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 az

$y_{11}, \ldots, y_{1n_1} := f_1(x_1, \ldots, x_m)$
$\vdots$
$y_{k1}, \ldots, y_{kn_k} := f_k(x_1, \ldots, x_m)$

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. Az f függvény g szerinti rekurziója által meghatározott feladat specifikációja:

$$A = \underset{x_1}{\mathbb{N}} \times \ldots \times \underset{x_{n+1}}{\mathbb{N}} \times \underset{y}{\mathbb{N}}$$
$$B = \underset{x'_1}{\mathbb{N}} \times \ldots \times \underset{x'_{n+1}}{\mathbb{N}}$$
$$Q : (x_1 = x'_1 \wedge \cdots \wedge x_{n+1} = x'_{n+1} \wedge (x_1, \ldots, x_{n+1}) \in \mathcal{D}_{\varrho(f,g)})$$
$$R : (Q \wedge y = \varrho(f, g)(x'_1, \ldots, x'_{n+1}))$$

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:

$$Q' : (Q \wedge k = 1 \wedge y = f(x_1, \ldots, x_n))$$

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 terminálófüggvénynek.

Az ötödik pont azt írja le, hogy az imént definiált terminálófüggvény értékének a ciklusmagban csökkennie kell. Ez elérhető a k eggyel való növelésével.

A négyes pont kielégítéséhez vizsgáljuk meg, mi lesz a leggyengébb előfeltétele a k-t növelő értékadásnak a P-re vonatkozóan.

$$\begin{aligned} Q'' = lf(k := k+1, P) \quad = \quad & (Q \wedge k+1 \in [1..x_{n+1}] \ \wedge \\ & y = \varrho(f,g)(x'_1, \ldots, x'_n, k+1)) \end{aligned}$$

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 levezetési szabálya, valamint a specifikáció tétele garantálja, hogy a

$k, y := 1, f(x_1, \ldots, x_n)$
$k \neq x_{n+1}$
$y := g(x_1, \ldots, x_n, k, y)$
$k := k + 1$

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:

$$\begin{aligned} A &= \underset{x_1}{\mathbb{N}} \times \ldots \times \underset{x_n}{\mathbb{N}} \times \underset{y}{\mathbb{N}} \\ B &= \underset{x'_1}{\mathbb{N}} \times \ldots \times \underset{x'_n}{\mathbb{N}} \\ Q &: (x_1 = x'_1 \wedge \cdots \wedge x_n = x'_n \wedge (x_1, \ldots, x_{n+1}) \in \mathcal{D}_{\mu(f)}) \\ R &: (Q \wedge y = \mu(f)(x'_1, \ldots, x'_n)) \end{aligned}$$

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. Egyszerűen belátható, hogy ennek a programnak a P-re vonatkozó leggyengébb előfeltétele,

$$\begin{aligned} lf(z, y := f(x_1, \ldots, x_n, 1), 1; P) \quad = \quad & (Q \wedge f(x_1, \ldots, x_n, 1) = f(x_1, \ldots, x_n, 1) \\ & \wedge \forall i \in [1..1-1] : f(x_1, \ldots, x_n, i) \neq 1) \end{aligned}$$

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 terminálófüggvényt: $t = N - y$. Ez kielégíti a levezetési szabály harmadik pontját.

Az ötödik pont megkívánja, hogy a terminálófüggvény 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

$z, y := f(x_1, \ldots, x_n, 1), 1$
$z \neq 1$
$z := f(x_1, \ldots, x_n, y+1)$
$y := y + 1$

program megoldja a $\mu(f)$ által meghatározott feladatot, azaz kiszámítja $\mu(f)$-et.

Következmény. Az előzőekben megmutattuk, hogy ha az alapfüggvények kiszámíthatók, akkor a belőlük – a parciális rekurzív függvényeknél megengedett – operátorokkal felépített függvények is kiszámíthatók jól konstruált programokkal. Ennek alapján kimondhatjuk az alábbi tételt:

9.1. TÉTEL: STRUKTURÁLT PROGRAMOZÁS ÉS KISZÁMÍTHATÓSÁG
Minden kiszámítható függvény kiszámítható egy jól konstruált programmal.

9.3. Relációk

Fordítsuk most figyelmünket a relációk felé. Ahhoz, hogy a relációk kiszámíthatóságát megvizsgálhassuk, definiálnunk kell a kiszámítható reláció fogalmát.

9.2. 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, minden kiszámítható (rekurzívan felsorolható) feladat – emlékezzünk, hogy minden feladat egy reláció – megoldható strukturált programmal, először megadjuk a rekurzívan felsorolható relációk egy másik jellemzését.

9.2. 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 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. Specifikáljuk a feladatot az alábbi módon:

$$A = \underset{x}{\mathbb{N}^k} \times \underset{y}{\mathbb{N}^k}$$

$$B = \underset{x'}{\mathbb{N}^k}$$

$$Q : (x = x' \wedge x \in \mathcal{D}_F)$$

$$R : (Q \wedge (x, y) \in F)$$

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:

$i, (z, y) := 1, \varphi(1)$	
$z \neq x$	
	$(z, y) := \varphi(i + 1)$
	$i := i + 1$

A bizonyításban a terminálófüggvény megadásánál ugyanazt a technikát alkalmazzuk, mint a μ-operátornál.

Ezzel az előzőleg függvényekre vonatkozó tételünket általánosíthatjuk relációkra:

9.3. TÉTEL: STRUKTURÁLT PROGRAMOZÁS ÉS KISZÁMÍTHATÓ RELÁCIÓK
Minden kiszámítható reláció kiszámítható egy jól konstruált programmal.

Megjegyezzük, hogy ezek az eredmények kiterjeszthetők a relatíve kiszámítható függvények (ugyanezen operátorokkal egy tetszőleges alaphalmazból képzett függvények) vagy a parciális rekurzív funkcionálok halmazára is [Odi 98].

10. 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.

10.1. A típusspecifikáció

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

10.1. DEFINÍCIÓ: TÍPUSSPECIFIKÁCIÓ

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

1. H az alaphalmaz,
2. $I_s : H \to \mathbb{L}$ a specifikációs invariáns,
3. $T_{\mathcal{T}} = \{(\mathcal{T}, x) \mid x \in \lceil I_s \rceil\}$ a *típusértékhalmaz*,
4. $\mathbb{F} = \{F_1, F_2, \ldots, F_n\}$ a típusműveletek specifikációja, ahol $\forall i \in [1..n]$: $F_i \subseteq A_i \times A_i$, $A_i = A_{i_1} \times \cdots \times A_{i_{n_i}}$ úgy, hogy $\exists j \in [1..n_i] : A_{i_j} = T_{\mathcal{T}}$.

Az alaphalmaz és az invariáns tulajdonság segítségével azt fogalmazzuk meg, hogy mi az a halmaz, $T_{\mathcal{T}}$, amelynek elemeivel foglalkozni akarunk, míg a feladatok halmazával azt írjuk le, hogy ezekkel az elemekkel 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.

10.2. A típus

Vizsgáljuk meg, hogy a típusspecifikációban leírt követelményeket hogyan valósítjuk meg.

10.2. DEFINÍCIÓ: TÍPUS

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

1. $\varrho \subseteq E^* \times T$ a reprezentációs függvény (reláció),
   T a típusértékhalmaz,
   E az elemi típusértékhalmaz,
2. $I : E^* \to \mathbb{L}$ típusinvariáns,
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, $B_i = B_{i_1} \times \cdots \times B_{i_{m_i}}$ 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 típusértékek reprezentációját írja le, míg a programhalmaz a típusműveletek implementációját tartalmazza. Az elemi típusértékhalmaz lehet egy tetszőleges másik típus típusértékhalmaza vagy egy, valamilyen módon definiát, legfeljebb megszámlálható halmaz.

Elemi típusnak nevezünk egy $\mathcal{T} = (\varrho, I, \mathbb{S})$ típust, ha $T = E$, és $\varrho|_{\lceil I \rceil} = Id_E$.

Meg kell még adnunk, hogy egy típus mikor felel meg a típusspecifikációnak, azaz mikor teljesíti a specifikációban leírt követelményeket.

Legyen a továbbiakban $\mathcal{T}_s = (H, I_s, \mathbb{F})$, $\mathcal{T} = (\varrho, I, \mathbb{S})$, $F \in \mathbb{F}$, és $S \in \mathbb{S}$, $F \subseteq A \times A$, $A = A_1 \times \cdots \times A_p$, $S \subseteq B \times B^{**}$, $B = B_1 \times \cdots \times B_q$.

Legyen A altere C-nek és B altere D-nek.

Azt mondjuk, hogy a $C = C_1 \times \cdots \times C_r$ és $D = D_1 \times \cdots \times D_r$ állapotterek *illeszkednek*, ha

$$\forall i \in [1..r] : D_i = \begin{cases} E^*, & \text{ha } C_i = T_{\mathcal{T}}; \\ C_i & \text{különben.} \end{cases}$$

Ebben az esetben legyen a $\gamma \subseteq D \times C$ reláció a következő:

$$\forall d \in D : \gamma(d) = \gamma_1(d_1) \times \gamma_2(d_2) \times \ldots \times \gamma_n(d_r),$$

ahol $\forall i \in [1..r] : \gamma_i \subseteq D_i \times C_i$ és

$$\gamma_i = \begin{cases} \varrho|_{\lceil I \rceil}, & \text{ha } C_i = T_{\mathcal{T}}; \\ id_{D_i} & \text{különben.} \end{cases}$$

Vegyük észre, hogy a γ tulajdonképpen a ϱ egyfajta kiterjesztése több komponensű, de egymáshoz illeszkedő állapotterek esetére.

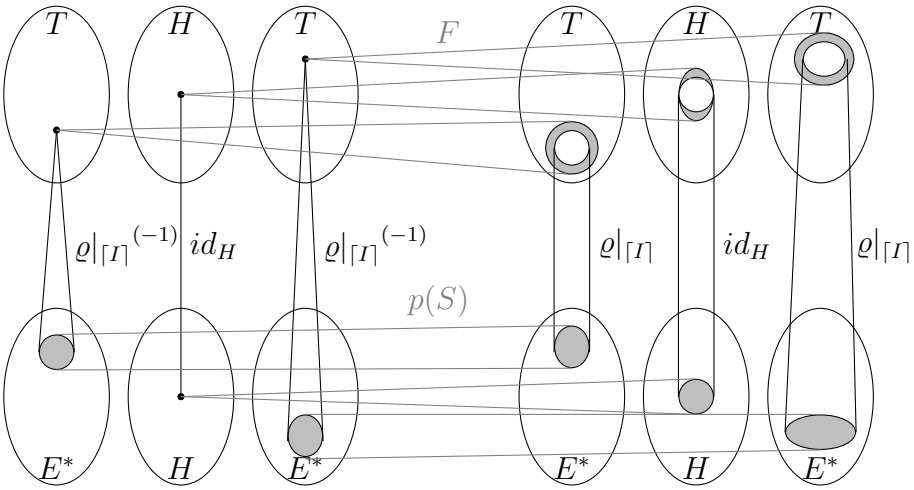
10.3. DEFINÍCIÓ: MEGOLDÁS ϱ-N KERESZTÜL

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, amelynek altere A, illetve B, hogy S' γ szerint megoldása F'-nek, ahol $\gamma \subseteq D \times C$ a fenti értelemben definiált leképezés, S' az S kiterjesztése D-re, F' pedig az F kiterjesztése C-re.

A kiterjesztési tételek szerint, ha vannak a definíciónak megfelelő illeszkedő terek, akkor mindegyik ilyen.

10.4. DEFINÍCIÓ: MEGFELELÉS

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



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

1. $\varrho(\lceil I \rceil) = T_{\mathcal{T}}$,
2. $\forall F \in \mathbb{F} : \exists S \in \mathbb{S} : S$ a ϱ-n keresztül megoldja F-et.

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

10.3. A típusspecifikáció tétele

A típusspecifikáció és a típus fogalmának bevezetésével tulajdonképpen a feladat fogalmát általánosítottuk, míg a megfeleltetés a megoldás fogalmának volt egyfajta általánosítása. A specifikáció tétele a megoldásra adott elégséges feltételt. Most a ϱ-n keresztüli megoldásra adunk egy hasonló feltételt.

10.1. TÉTEL: TÍPUSSPECIFIKÁCIÓ TÉTELE

Legyen $\mathcal{T}_s = (H, I_s, \mathbb{F})$ és $\mathcal{T} = (\varrho, I, \mathbb{S})$ adott típusspecifikáció és típus, és tegyük fel, hogy a reprezentáció helyes, azaz $\varrho(\lceil I \rceil) = T_{\mathcal{T}}$. 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 reláció szerinti megoldás tétele miatt csak azt kell belátnunk hogy $\mathcal{D}_F \subseteq \mathcal{R}_\gamma$ teljesül. Ez pedig nyilvánvaló, mivel föltettük, hogy $\varrho(\lceil I \rceil) = T_{\mathcal{T}}$.

□

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).

10.4. Absztrakt típus

Ebben a részben a megfelelés fogalmának általánosításával foglalkozunk. Először definiáljuk az ekvivalens feladat fogalmát. Ez az 5. fejezet alapján eléggé kézenfekvő. Az 5.2.-ben bevezetett jelöléseket használjuk.

10.5. DEFINÍCIÓ: FELADATOK EKVIVALENCIÁJA

Az $F_1 \subseteq A \times A$ és az $F_2 \subseteq B \times B$ feladatot *ekvivalensnek* nevezzük, ha $A \overset{f}{\sim} B$ és $\gamma_f(F_2) = F_1$.

Most megadjuk két típusspecifikáció ekvivalenciájának definícióját. Két típusspecifikációt ekvivalensnek nevezünk, ha lényegében csak a nevükben különböznek, azaz:

10.6. DEFINÍCIÓ: TÍPUSSPECIFIKÁCIÓK EKVIVALENCIÁJA

A $\mathcal{T}_{s1} = (H_1, I_{s1}, \mathbb{F}_1)$ és a $\mathcal{T}_{s2} = (H_2, I_{s2}, \mathbb{F}_2)$ *típusspecifikáció ekvivalens*, ha $\lceil I_{s_1} \rceil = \lceil I_{s_2} \rceil$, és létezik $\xi : \mathbb{F}_1 \to \mathbb{F}_2$ bijekció, hogy $\forall F \in \mathbb{F}_1 : \xi(F) \sim F^{\mathcal{T}_2 \leftarrow \mathcal{T}_1}$, ahol

$F^{\mathcal{T}_2 \leftarrow \mathcal{T}_1} = \{(x^{\mathcal{T}_2 \leftarrow \mathcal{T}_1}, y^{\mathcal{T}_2 \leftarrow \mathcal{T}_1}) \mid (x, y) \in F\}$, ahol

$x^{\mathcal{T}_2 \leftarrow \mathcal{T}_1} = \{(i, a^{\mathcal{T}_2 \leftarrow \mathcal{T}_1}) \mid (i, a) \in x\}$, ahol

$$a^{\mathcal{T}_2 \leftarrow \mathcal{T}_1} = \begin{cases} (\mathcal{T}_1, e), & \text{ha } a = (\mathcal{T}_2, e); \\ a & \text{különben.} \end{cases}$$

Ezután a megfelelés fogalmát általánosítjuk.

10.7. DEFINÍCIÓ: MEGFELELÉS ÁLTALÁNOSÍTÁSA

Azt mondjuk, hogy a $\mathcal{T} = (\varrho, I, \mathbb{S})$ típus – általános értelemben – *megfelel* a $\mathcal{T}_s = (H, I_s, \mathbb{F})$ típusspecifikációnak, ha létezik olyan ekvivalens típusspecifikáció, aminek az eredeti értelemben megfelel.

A típusértékhalmazokat, mint párok halmazait definiáltuk. A fenti definíció értelmében azonban, a megfelelés szempontjából, az első komponensnek – a névnek – nincs jelentősége, ezért azzal a továbbiakban nem is foglalkozunk.

A programfüggvény általánosításaként definiálhatjuk az absztrakt típust:

10.8. DEFINÍCIÓ: ABSZTRAKT TÍPUS

Legyen $\mathcal{T} = (\varrho, I, \mathbb{S})$ egy típus. A $p(\mathcal{T}) = (p(\varrho), p(\mathbb{S}))$-t $\mathcal{T}$ *absztrakt típusának* nevezzük, ha

1. $\forall \varepsilon \in E^* : p(\varrho)(\varepsilon) = \varrho|_{\lceil I \rceil}(\varepsilon)_2$,
2. $p(\mathbb{S}) = \{p(S) \mid S \in \mathbb{S}\}$.

A programokhoz hasonlóan a $\mathcal{T}_1 = (\varrho_1, I_1, \mathbb{S}_1)$ és a $\mathcal{T}_2 = (\varrho_2, I_2, \mathbb{S}_2)$ típust *ekvivalensnek* nevezzük, ha $p(\mathcal{T}_1) = p(\mathcal{T}_2)$. Az elnevezés azért indokolt, mert a megfelelés szempontjából az ekvivalens típusok egyformák.

10.5. Példák

10.1. példa: A típusértékek halmaza legyen a magyar ábécé magánhangzói: { a, á, e, é, i, í, o, ó, ö, ő, u, ú, ü, ű }. Szeretnénk tudni, hogy egy adott magánhangzónak melyik a (rövid, illetve 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! Adjon típusspecifikációt, és készítsen 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 $F \subseteq T_{\mathcal{T}} \times T_{\mathcal{T}}$,

$$F = \begin{array}{l}\{\text{ (a, á), (á, a), (e, é), (é, e), (i, í), (í, i), (o, ó),}\\ \text{(ó, o), (ö, ő), (ő, ö), (u, ú), (ú, u), (ü, ű), (ű, ü) }\}.\end{array}$$

Felhívjuk a figyelmet arra, hogy $T_{\mathcal{T}}$ elemeit az egyszerűség kedvéért $(\mathcal{T}, x)$ helyett x-el jelöltük. Továbbiakban is ezt a megoldást fogjuk alkalmazni. Adjuk meg a típust!
$\mathcal{T} = (\varrho, I, \{S\})$, ahol $\varrho \subseteq E^* \times MGH$,

$$\varrho = \begin{array}{l}\{(<0>, \text{a}), (<14>, \text{á}), (<1>, \text{e}), (<13>, \text{é}), (<2>, \text{i}),\\ (<12>, \text{í}), (<3>, \text{o}), (<11>, \text{ó}), (<4>, \text{ö}), (<10>, \text{ő}),\\ (<5>, \text{u}), (<9>, \text{ú}), (<6>, \text{ü}), (<8>, \text{ű}) \};\end{array}$$

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

$$I(\alpha) = \begin{cases} igaz, & \text{ha } |\alpha| = 1 \text{ és } \alpha_1 \neq 7; \\ hamis & \text{egyébként.} \end{cases}$$

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

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

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 típusinvariánst és a programot is.

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

$$\begin{array}{rcl} H & = & \wp([0..127]), \\ I_s & = & Igaz, \\ \mathbb{F} & = & \{F_m, F_u, F_e\}, \end{array}$$

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

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

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

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

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

$$F_e = \{((h,e,l),(h',e',l')) \mid h = h' \text{ és } e = e' \text{ és } l' = (e \in h)\}.$$

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

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

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

$$I(\alpha) = \begin{cases} igaz, & \text{ha } |\alpha| = 128; \\ hamis & \text{egyébként.} \end{cases}$$

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

$$\begin{aligned} p(S_m) \quad = \quad & \{((\alpha,\beta,\gamma),(\alpha',\beta',\gamma'))|\alpha \in \lceil I \rceil \text{ és } \beta \in \lceil I \rceil \text{ és } \gamma' \in \lceil I \rceil \text{ és } \alpha = \alpha' \text{ és} \\ & \beta = \beta' \text{ és } \forall i \in [1..128] : \gamma'_i = \alpha_i \cdot \beta_i\}; \end{aligned}$$

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

$$\begin{aligned} p(S_u) \quad = \quad & \{((\alpha,\beta,\gamma),(\alpha',\beta',\gamma'))|\alpha \in \lceil I \rceil \text{ és } \beta \in \lceil I \rceil \text{ és } \gamma' \in \lceil I \rceil \text{ és } \alpha = \alpha' \text{ és} \\ & \beta = \beta' \text{ és } \forall i \in [1..128] : \gamma'_i = \alpha_i + \beta_i - \alpha_i \cdot \beta_i\}; \end{aligned}$$

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

$$p(S_e) = \{((\alpha,x,l),(\alpha',x',l')) \mid \alpha \in \lceil I \rceil \text{ és } \alpha = \alpha' \text{ és } x = x' \text{ és } l' = (\alpha_{x+1} = 1)\}.$$

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

$$\varrho(\lceil I \rceil) = [I_s].$$

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.

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

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

10.3. példa: Legyen $\mathcal{T}_s = (H, I_s, \mathbb{F})$ 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.

10.6. Feladatok

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

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

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

Készítsen egy olyan típust, amely megfelel a specifikációnak!

10.3. Legyen $\mathcal{T}_{s1} = (H_1, I_{s1}, \mathbb{F}_1), \mathcal{T}_{s2} = (H_2, I_{s2}, \mathbb{F}_2)$ 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?

10.4. Adott a $\mathcal{T}_s = (H, I_s, \mathbb{F})$ 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?

10.5. Legyen $\mathcal{T}_s = (H, I_s, \mathbb{F})$ 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?

10.6. Legyen $\mathcal{T}_S = (T, I_S, \mathbb{F})$ a következő típusspecifikáció:
$I_S = Igaz, T_2 = \mathbb{N}_0, \mathbb{F} = \{F_1, F_2\}$.
F_1 specifikációja:

$$\begin{array}{lll} A = & T & \\ & x & \\ B = & T & \\ & x' & \end{array}$$

$Q : (x = x')$

$R : (\exists z \in \mathbb{Z} : \ x'_2 = 8 \cdot z + x_2 \text{ és } 0 \leq x_2 < 8)$

F_2 specifikációja:

$$\begin{array}{lllllll} A = & T & \times & T & \times & \mathbb{L} & \\ & x & & y & & l & \\ B = & T & \times & T & & & \\ & x' & & y' & & & \end{array}$$

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

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

$\mathcal{T} = (\varrho, I, \S), E = \{0,1,2,3,4,5,6,7\}$,

$\forall e \in E^* : \ \varrho(e) = \{(\mathcal{T}, \sum_{i=1}^{|e|} (e_i \cdot 8^{|e|-i})\}$

a)

$$I(e) = \begin{cases} igaz, & \text{ha } |e| \geq 1 \text{ és } (e_1 = 0 \Rightarrow |e| = 1); \\ hamis & \text{egyébként} \end{cases} .$$

b)

$$I(e) = \begin{cases} igaz, & \text{ha } |e| \geq 1; \\ hamis & \text{egyébként} \end{cases} .$$

$S_1 \subseteq (E^*) \times (E^*)^{**}$

$$\begin{aligned} \forall e \in E^* :\ S_1(e) = \{\alpha \in (E^*)^* \mid\ & |\alpha| = |e| \text{ és} \\ & \forall i \in [1..|\alpha|] :\ |\alpha_i| = |\alpha| - i + 1 \text{ és} \\ & \forall i \in [2..|\alpha|] :\ \forall j \in [1..|\alpha_i|] : \alpha_{i_j} = \alpha_{i-1_{j+1}})\} \end{aligned}$$

$S_2 \subseteq (E^* \times E^* \times \mathbb{L}) \times (E^* \times E^* \times \mathbb{L})^{**}$
$\forall e, d \in E^* :\ \forall l \in \mathbb{L} :$

$$\begin{aligned} S_2(e,d,l) = \{\beta \in (E^* \times E^* \times \mathbb{L}) \times (E^* \times E^* \times \mathbb{L})^{**} \mid\ & \\ |\beta| = \min(|e|,|d|) + 1 \text{ és} & \\ \forall i \in [2..|\beta|] :\ \beta_i = (ee, dd, ll) \text{ és} & \\ ll = (\forall j \in [1..i-1] :\ ee_j = dd_j) \text{ és} & \\ |ee| = i - 1 \text{ és } |dd| = i - 1) \text{ és} & \\ \forall j \in [1..i-1] :\ (ee_{i-j} = e_{|e|-j+1} \text{ és } dd_{i-j} = d_{|d|-j+1}))\} & \end{aligned}$$

Írja le szavakkal az F_1, F_2 feladatot, a ϱ relációt és az S_1, S_2 programfüggvényét! Megfelel-e a típus a specifikációnak az a), illetve a b) esetben?

10.7. Adjunk típusspecifikációt, reprezentációs függvényt, absztrakt típust (ami megfelel a specifikációnak)! A típusértékek: a síkvektorok halmaza, a műveletek: két vektor összeadása, valamint annak eldöntése, hogy két vektor számszorosa-e egymásnak.

10.8. Adjunk típusspecifikációt, reprezentációs függvényt, absztrakt típust (ami megfelel a specifikációnak)! A típusértékek: a térvektorok halmaza, a műveletek: két vektor kivonása, valamint egy vektornak egy számmal való szorzása.

10.9. Adjunk típusspecifikációt, reprezentációs függvényt, absztrakt típust (ami megfelel a specifikációnak) a komplex számok típusára, ahol a műveletek két komplex szám összeadása és egy komplex szám képzetes részének meghatározása.

10.10. Adjunk típusspecifikációt, reprezentációs függvényt, absztrakt típust (ami megfelel a specifikációnak) a komplex számok típusára, ahol a műveletek két komplex szám összeszorzása és egy komplex szám n-dik ($n \in \mathbb{N}$) hatványának meghatározása.

10.11. Adjunk típusspecifikációt, reprezentációs függvényt, absztrakt típust (ami megfelel a specifikációnak). A típusértékek a körlemezek halmaza, a műveletek: egy körlemez eltolása és annak eldöntése, hogy egy síkbeli pont rajta van-e a körlemezen.

10.12. Adjunk típusspecifikációt, reprezentációs függvényt, absztrakt típust (ami megfelel a specifikációnak). A típusértékek: a gömbök halmaza, a műveletek: egy gömb eltolása és annak eldöntése, hogy egy térbeli pont benne van-e a gömbben.

10.13. Adjunk típusspecifikációt, reprezentációs függvényt, absztrakt típust (ami megfelel a specifikációnak). A típusértékek: a négyzetek halmaza, a műveletek: egy négyzet eltolása, egy négyzet méretének megváltoztatása, egy négyzet területének kiszámítása és annak eldöntése, hogy egy síkbeli pont rajta van-e a négyzeten.

10.14. Adjunk típusspecifikációt, reprezentációs függvényt, absztrakt típust (ami megfelel a specifikációnak). A típusértékek: a kockák halmaza, a műveletek: egy kocka eltolása, egy kocka méretének megváltoztatása, egy kocka térfogatának kiszámítása és annak eldöntése, hogy egy térbeli pont benne van-e a kockában.

11. fejezet

Típuskonstrukciók

Azzal már foglalkoztunk, 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.

A legkézenfekvőbb lehetőség új típus létrehozására, hogy egy adott típus műveleteinek halmazát megváltoztatjuk, új műveletet veszünk hozzá, elhagyunk belőle, megváltoztatunk meglevő műveletet. Ez egy egyszerű és nagyon gyakran alkalmazott eljárás.

A másik, egy kicsit bonyolultabb lehetőség az invariáns tulajdonság megváltoztatása. Az invariáns szűkítése könnyen kezelhető és jól használható lehetőség.

A harmadik lehetőség a reprezentációs függvény megváltoztatása. Ez is egy fontos lehetőség. A továbbiakban ennek általánosabb esetével fogunk foglalkozni, meglevő reprezentációs függvényekből állítunk elő újakat.

11.1. A megengedett konstrukciók

Természetesen sokféle lehetőségünk van meglevő reprezentációs függvényekből újat csinálni, de mi a továbbiakban csak három speciális konstrukcióval foglalkozunk: 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, $E_1, E_2, \ldots, E_n$-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.

11.1. DEFINÍCIÓ: DIREKTSZORZAT

A $\mathcal{T} = (\varrho, I, \mathbb{S})$ típus *direktszorzata* a $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n$ típusoknak, ha

$$\varrho = \varphi_{\mathfrak{D}} \circ \psi_{\mathfrak{D}}$$

ahol $\varphi_{\mathfrak{D}} \subseteq B \times T$, $\psi_{\mathfrak{D}} \subseteq E^* \times B$ és

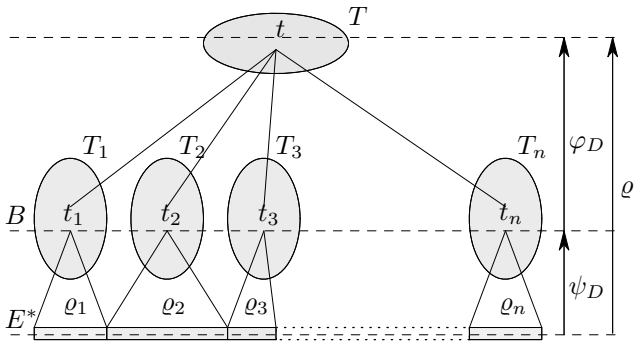
$$\psi_{\mathfrak{D}} = \{(\varepsilon, b) \in E^* \times B \mid \forall i \in [1..n] : \exists \varepsilon_i \in E_i^* : (\varepsilon_i, b_i) \in \varrho_i \ \wedge \ \varepsilon = kon(\varepsilon_1, \ldots, \varepsilon_n)\}.$$

A direktszorzat értékhalmazára bevezetjük a $T = (T_1, T_2, \ldots, T_n)$ jelölést.

Ha $\varphi_{\mathfrak{D}}$ kölcsönösen egyértelmű leképezés, akkor a direktszorzatot *rekord* típusnak nevezzük. A direktszorzat-típusok általában rekordok, de nem mindig. Például tekintsük a racionális számok halmazának egy lehetséges reprezentációját:
$B = \mathbb{Z} \times \mathbb{Z}$, $\varphi_{\mathfrak{D}} \subseteq B \times \mathbb{Q}$:

$$((x, y), t) \in \varphi_{\mathfrak{D}} \iff y \neq 0 \wedge t = x/y$$

Egyszerűen látható, hogy a fent definiált $\varphi_{\mathfrak{D}}$ reláció a racionális számok halmazát reprezentálja, de nem kölcsönösen egyértelmű.



11.1. 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 $\varphi_{\mathfrak{D}}$ 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 $\varphi_{\mathfrak{D}}$ 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.

11.2. 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

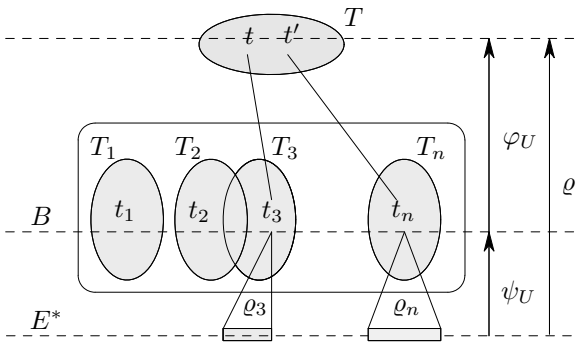
$$\varrho = \varphi_{\mathfrak{U}} \circ \psi_{\mathfrak{U}},$$

ahol $\varphi_{\mathfrak{U}} \subseteq B \times T$, $\psi_{\mathfrak{U}} \subseteq E^* \times B$ és

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

Az unió értékhalmazának jelölése: $T = (T_1; T_2; \ldots; T_n)$.

Itt is külön elnevezést adtunk annak az esetnek, amikor a $\varphi_{\mathfrak{U}}$ leképezés kölcsönösen egyértelmű, ekkor az uniót *egyesítésnek* nevezzük.



11.2. ábra. Unió konstrukció

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, E a $\mathcal{T}_0$ típus elemi típusértékhalmaza és $B = T_0^*$.

11.3. DEFINÍCIÓ: ITERÁLT

Azt mondjuk, hogy a $\mathcal{T} = (\varrho, I, \mathbb{S})$ típus *iteráltja* a $\mathcal{T}_0$ típusnak, ha

$$\varrho = \varphi_{\mathfrak{I}} \circ \psi_{\mathfrak{I}},$$

ahol $\varphi_{\mathfrak{I}} \subseteq B \times T$, $\psi_{\mathfrak{I}} \subseteq E^* \times T_0^*$ és

$$\begin{aligned} \psi_{\mathfrak{I}} \quad = \quad & \{(\varepsilon, b) \in E^* \times B \mid \exists \varepsilon_1, \ldots, \varepsilon_{|b|} \in E^* : (\varepsilon_i, b_i) \in \varrho_0 \ \wedge \\ & \varepsilon = kon(\varepsilon_1, \ldots, \varepsilon_{|b|})\}. \end{aligned}$$

Az iterált típusértékhalmazának jelölése: $T = it(T_0)$.

Az iterált típuskonstrukciónak három speciális esetét különböztetjük meg aszerint, hogy a $\varphi_{\mathfrak{I}}$ leképezésre milyen feltételek teljesülnek.

- Ha a $\varphi_{\mathfrak{I}}$ leképezés kölcsönösen egyértelmű, akkor *sorozat* típuskonstrukcióról beszélünk, és típusértékhalmazát $T = seq(T_0)$-lal jelöljük.
- Ha

  $$(\alpha, t), (\beta, t) \in \varphi_{\mathfrak{I}} \Leftrightarrow \alpha \in perm(\beta),$$

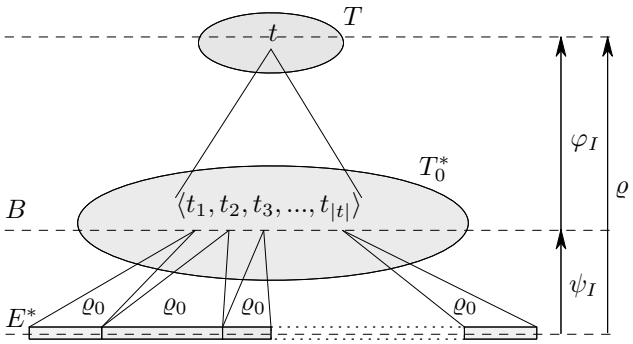
  akkor az iterált konstrukciót *kombináció* típusnak nevezzük. A kombináció értékhalmazának jelölése: $T = com(T_0)$.

- Ha

$$(\alpha,t),(\beta,t)\in\varphi_{\mathfrak{I}}\Leftrightarrow\bigcup_{i=1}^{|\alpha|}\{\alpha_i\}=\bigcup_{i=1}^{|\beta|}\{\beta_i\},$$

  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.



11.3. ábra. Iterált konstrukció

A programokhoz hasonlóan, ha mást nem mondunk, *primitív típusnak* fogjuk tekinteni azokat az elemi típusokat, amiknek a típusértékhalmaza a természetes számok, az egész számok, a logikai értékek vagy a karakterek.

11.2. Szelektorfüggvények

Az előzőekben definiált típuskonstrukciókra most bevezetünk néhány olyan függvényt és jelölést, amelyek leegyszerűsítik a rájuk vonatkozó állítások, programok megfogalmazását.

Legyen $T=(T_1,T_2,\ldots,T_n)$ rekord. A $\varphi_{\mathfrak{D}}^{(-1)}$ 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

$$\forall t\in T:\varphi_{\mathfrak{D}}(s_1(t),s_2(t),\ldots,s_n(t))=t.$$

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 $\varphi_{\mathfrak{U}}$ 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 $\forall i \in [1..n]$: $\forall t \in T$:

$$s_i(t) = \left(\varphi_{\mathfrak{U}}^{(-1)}(t) \in T_i\right).$$

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 $\forall t \in T : \forall i \in [1..|\varphi_{\mathfrak{I}}^{(-1)}(t)|]$:

$$s(t, i) = \varphi_{\mathfrak{I}}^{(-1)}(t)_i.$$

A sorozat szelektorát nem szoktuk külön elnevezni, helyette indexelést alkalmazunk, azaz a $t_i = s(t, i)$ jelölést használjuk.

11.3. Az iterált specifikációs függvényei

Ha az iterált típus az előzőekben bevezetett három speciális osztály valamelyikébe tartozik, akkor további függvényeket definiálunk hozzá.

Legyen $T = it(T_0)$, $(\alpha, t) \in \varphi_{\mathfrak{I}}$, 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 jól definiált, 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 $T = seq(T_0)$, $(\alpha, t) \in \varphi_{\mathfrak{I}}$, $\alpha = \langle \alpha_1, \alpha 2, \ldots \alpha_{|\alpha|} \rangle$.

- Nem üres sorozat első és utolsó eleme: $lov :\in T \to T_0$, $hiv :\in T \to T_0$,

$$\begin{aligned} lov(t) &= \alpha_1; \\ hiv(t) &= \alpha_{|\alpha|}. \end{aligned}$$

- Sorozat kiterjesztése a sorozat elején vagy végén (legyen $e \in T_0$): $loext : T \times T_0 \to T$, $hiext : T \times T_0 \to T$,

$$\begin{aligned} loext(t, e) &= \varphi_{\mathfrak{I}}(con(\langle e \rangle, \alpha)); \\ hiext(t, e) &= \varphi_{\mathfrak{I}}(con(\alpha, \langle e \rangle)). \end{aligned}$$

- Nem üres sorozat első vagy utolsó elemének elhagyásával kapott sorozat: $lorem :\in T \to T$, $hirem :\in T \to T$,

$$\begin{aligned} lorem(t) &= \varphi_{\mathfrak{I}}(\langle \alpha_2, \ldots, \alpha_{|\alpha|} \rangle); \\ hirem(t) &= \varphi_{\mathfrak{I}}(\langle \alpha_1, \ldots, \alpha_{|\alpha|-1} \rangle). \end{aligned}$$

11.4. 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.

11.4. DEFINÍCIÓ: FÜGGVÉNYTÍPUS
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 $F = fun(H, E),\ ((h, t), f) \in \varphi_{\mathfrak{D}}$. Ekkor

- $dom : F \to \mathbb{N}_0$,
$$dom(f) = dom(t).$$
- $lob : F \to H$,
$$lob(f) = h.$$
- $hib : F \to H$,
$$hib(f) = succ^{dom(f)-1}(h).$$
- $lov :\in F \to E$,
$$lov(f) = lov(t).$$
- $hiv :\in F \to E$,
$$hiv(f) = hiv(t).$$
- $loext, hiext : F \times E \to F$,
$$\begin{aligned} loext(f, e) &= \varphi_{\mathfrak{D}}(pred(h), loext(t, e)); \\ hiext(f, e) &= \varphi_{\mathfrak{D}}(h, hiext(t, e)). \end{aligned}$$
- $lorem, hirem :\in F \to F$,
$$\begin{aligned} lorem(f) &= \varphi_{\mathfrak{D}}(succ(h), lorem(t)); \\ hirem(f) &= \varphi_{\mathfrak{D}}(h, hirem(t)). \end{aligned}$$

A sorozathoz hasonlóan a függvénytípusra is bevezetünk egy szelekciós parciális függvényt. Tekintsük a fentiekben használt f-et. Ekkor $s_f :\in H \to E$, $\mathcal{D}_{s_f} = \{succ^i(lob(f)) \mid 0 \le i < dom(f)\}$, 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, vagy egyszerűen indexelünk, azaz

$$f_g = f(g) = s_f(g).$$

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”, értékeit pedig a sorozatkomponens tartalmazza.

Az előbbiekben bevezetett dom, lov, hiv, lob, hib függvényeket kiterjeszthetjük az egész állapottérre is: komponáljuk a megfelelő változóval. Tehát ha például x egy sorozat típusú 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 a $t \circ x$ helyett $x.t$-t írunk.

11.5. A típuskonstrukciók típusműveletei

A típuskonstrukciók eddigi tárgyalásából még hiányzik valami: nem beszéltünk még a konstruált típusok műveleteiről. Az előbb felsorolt speciális esetekhez – az imént definiált 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 állapottéren értelmezett függvényeket kapunk. Az $s_i \circ t$ függvénykompozíciót a továbbiakban $t.s_i$-vel fogjuk jelölni. Egy rekord típusnál a szelektorfüggvény használatát megengedettnek tekintjük.

Ezenkívül 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 függvénynek ($t.s_i$) „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 $t := \varphi_{\mathfrak{U}}(t_i)$ 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 a következő 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 a sorozattípusokat osztályokba soroljuk a megengedett műveleteik alapján.

Legyen a továbbiakban $T = seq(E)$, $t : T$, $e : E$. Ekkor az iménti szelektorokhoz hasonlóan bevezetjük az alábbi jelöléseket:

$$\begin{aligned} dom \circ t &\to t.dom \\ lov \circ t &\to t.lov \\ hiv \circ t &\to t.hiv \end{aligned}$$

Természetesen $t.lov$ és $t.hiv$ csak parciális függvények. Ezenkívül 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 bal oldalon függvényként, a jobb oldalon pedig a művelet neveként használjuk. Lényeges ezért megjegyezni, hogy a jobb oldalon található műveletek csak a bal oldali értékadás egyszerűsítő *jelölései*.

Attól függően, hogy a fent definiált műveletek közül melyeket tekintjük megengedettnek, különböző konstrukciókról beszélünk.

Legyen $T = seq(E)$. Ekkor a T

- *szekvenciális input fájl*, ha csak a $lopop$ a megengedett művelet;
- *szekvenciális output fájl*, 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 fájl a $lopop$ művelettel használható legyen, tudnunk kell, hogy mikor olvastuk ki az utolsó elemet a fájlból. Ezt a problémát úgy szoktuk megoldani, hogy bevezetünk egy extremális elemet, és kikötjük, hogy a fájlban ez az utolsó elem (tehát még az üres fájl is tartalmazza). Ez a technika valósul meg azokban az operációs rendszerekben, ahol a szövegfájlok végét fájlvége (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 szimultán értékadást értjük:

$$sx, dx, x : read = \begin{cases} norm, lov(x), lorem(x), & \text{ha } dom(x) \neq 0; \\ abnorm, dx, x, & \text{ha } dom(x) = 0. \end{cases}$$

Ha egy szekvenciális fájlra a $read$ művelet van megengedve, akkor nincs szükség extremális elemre, helyette az sx változó értéke alapján lehet eldönteni, hogy végére értünk-e a fájlnak.

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 függvénytípus 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 végezhető el, ha $f.lob \leq i \leq f.hib$. Ekkor a fenti jelölésen azt az $f := f'$ értékadást értjük, amelyre:

$$f'.lob = f.lob \wedge f'.hib = f.hib \wedge f'(i) = e \wedge$$
$$\forall j \in [f.lob..f.hib] : j \neq i \rightarrow f'(j) = f(j).$$

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: $V = vekt(H, E)$.

12. fejezet

Programozási tételek (Levezetés)

Először néhány egyszerű feladatra keresünk megoldóprogramot. Ezek a feladatok két szempontból is fontosak számunkra. Egyrészt sok olyan konkrét feladat van, amelyeknek ezek általánosításai, így megoldásuk lehetőséget ad sok konkrét (konkrétabb) feladat megoldására is; ezért nevezzük őket programozási tételeknek. Másrészt megoldásukon keresztül megmutatjuk, hogyan lehet megoldóprogramot levezetni. Ezután néhány összetettebb tételt vezetünk le. Végül a tételek egy fontos csoportjával, speciális tulajdonságú függvények helyettesítési értékének kiszámításával foglalkozunk.

Ebben a fejezetben a feladatok megfogalmazásában használjuk a következő kijelentést: adott az $f : X \to Y$ függvény. Ezen azt fogjuk érteni, hogy a feladat A állapotterének van egy olyan H altere, amin az $X \to Y$ függvényeket definiáljuk, és a H altér változóit egy-egy megfelelő paraméterváltozóval az elő- és utófeltételben rögzítjük.

12.1. Programozási tételek intervallumon

A következő tételeknek sok közös vonása van. Mindegyik arról szól, hogy egy $[m..n]$ intervallumon teljesül egy φ tulajdonság. A φ az intervallumon kívül az állapottér számos más komponensétől is függhet.

$$
\begin{array}{lccccc}
A = & \mathbb{Z} & \times & \mathbb{Z} & \times & \dots \\
 & m & & n & & \dots \\
B = & \mathbb{Z} & \times & \mathbb{Z} & \times & \dots \\
 & m' & & n' & & \dots
\end{array}
$$

$$Q : (m = m' \wedge n = n' \wedge m \leq n(+1))$$

$$R : (Q \wedge \varphi(m, n, \dots))$$

Az, hogy az előfeltételben $n + 1$ vagy n szerepel, attól függ, hogy a feladatnak van-e értelme üres intervallum esetén, vagy nincs.

Az ilyen feladatokat ciklussal oldhatjuk meg. Invariáns tulajdonságnak azt választjuk, hogy φ nem az egész $[m..n]$ intervallumra, hanem csak egy $[m..k]$ részére teljesül, ezért az állapotteret kibővítjük egy új egész komponenssel (k), és kiterjesztjük

a feladatot az új állapottérre. Természetesen a kiterjesztésnek csak elvi jelentősége van, hiszen a kiterjesztett feladat specifikációja formálisan csak az állapottérben különbözik az eredetitől. Tehát az invariáns tulajdonság:

$$P = (Q \wedge k \in [m(-1)..n] \wedge \varphi(m, k, \dots))$$

Ez az invariáns tulajdonság azért megfelelő mert:

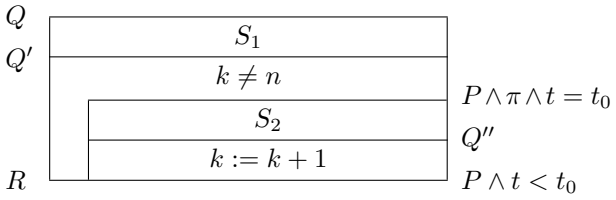
1. A ciklusfeltételt $k \neq n$-nek választva $P \wedge \neg\pi \Rightarrow R$, azaz teljesül a ciklus levezetési szabályának a 2. feltétele.

2. A levezetési szabály első feltétele általában nem teljesül, de könnyen tudunk olyan Q'-t választani, amiből már következik P, és könnyű eljutni Q-ból Q'-be. Ez a Q' rendszerint az üres, illetve néha az egy hosszúságú intervallum esete, azaz
$$Q' = (Q \wedge k = m(-1) \wedge \varphi(m, k, \dots)\,.$$

3. Még terminálófüggvényt kell választani, ilyen esetekben kézenfekvő választás:
$$t = (n - k)\,, \text{ hiszen így } P \wedge \pi \Rightarrow t > 0.$$

4. Most már a csak következő két feltételnek kell teljesülnie:
$$Q \Rightarrow lf(S_1, Q') \text{ és } P \wedge \pi \wedge t = t_0 \Rightarrow lf(S_0, P \wedge t < t_0).$$
Általánosságban még annyit mondhatunk, hogy a második esetben kézenfekvő, hogy S_0-t szekvencia formában keressük. Egyszerűen belátható, hogy a szekvencia második tagjaként alkalmazott
$$k := k + 1$$
értékadás csökkenti a terminálófüggvény értékét. Azért, hogy az invariáns tulajdonság is teljesüljön, a szekvencia közbülső tulajdonsága
$$Q'' = (lf(k := k + 1, P) \wedge t = t_0) = \left(P^{k \leftarrow k+1} \wedge t = t_0\right)$$
lesz, mivel így
$$Q'' \Rightarrow (lf(k := k + 1, P \wedge t < t_0)) = \left(P^{k \leftarrow k+1} \wedge n - k - 1 < t_0\right)$$
valóban teljesül.

A specifikáció tétele értelmében, ha S_1 és S_2 olyan programok, hogy $\forall b \in B$-re, vagy másképpen fogalmazva, a paraméterváltozók értékétől függetlenül
$$Q \Rightarrow lf(S_1, Q')$$
és
$$P \wedge \pi \wedge t = t_0 \Rightarrow lf(S_2, Q'' \wedge t = t_0)$$
teljesül, akkor a megoldóprogram sémája:



12.1.1. 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.

$$\begin{array}{lccccc} A = & \mathbb{Z} & \times & \mathbb{Z} & \times & \mathbb{Z} \\ & m & & n & & s \\ B = & \mathbb{Z} & \times & \mathbb{Z} & & \\ & m' & & n' & & \end{array}$$

$Q : (m = m' \wedge n = n' \wedge m \leq n + 1)$

$R : (Q \wedge s = \sum_{i=m}^{n} f(i))$

Ebben az esetben

$$\varphi(m, n, s) = \left(s = \sum_{j=m}^{n} f(j) \right),$$

így az invariáns tulajdonság:

$$P = \left(Q \wedge k \in [m-1..n] \wedge s = \sum_{j=m}^{k} f(j) \right),$$

a Q' állítás:

$$Q' = (Q \wedge k = m - 1 \wedge s = 0).$$

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

$Q \Rightarrow lf(k, s := m - 1, 0, Q') = (Q \wedge m - 1 = m - 1 \wedge 0 = 0) = Q.$

Már nincs más dolgunk, mint találni egy olyan programot, ami $P \wedge \pi \wedge t = t_0$-ból Q''-be jut, ahol

$$Q'' = \left(P^{k \leftarrow k+1} \wedge t = t_0\right) = \left(Q \wedge k + 1 \in [m-1..n] \wedge s = \sum_{j=m}^{k+1} f(j) \wedge t = t_0 \right).$$

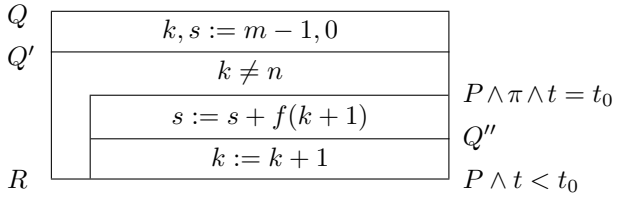
Nézzük meg, hogy mi nem teljesül Q''-ben: mivel $k \in [m-1..n]$ (P) és $k \neq n$ (π), $k+1 \in [m-1..n]$ fennáll. s értéke viszont nem jó, mert P szerint csak k-ig tartalmazza f értékeinek összegét, Q'' szerint pedig már $k+1$-ig kell. A fenti meggondolás alapján tehát s növelése $f(k+1)$-gyel jó lesz, azaz:

$P \wedge \pi \wedge t = t_0 \Rightarrow lf(s := s + f(k+1), Q'') =$

$$= \left(Q \wedge k + 1 \in [m-1..n] \wedge s + f(k+1) = \sum_{i=m}^{k+1} f(i) \wedge t = t_0 \right).$$

A fenti levezetés alapján kimondható az alábbi tétel:

Tétel: Az alábbi, struktogram formában megadott program megoldása a fent specifikált feladatnak:



12.1.2. 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.

$$\begin{array}{lccccc} A = & \mathbb{Z} & \times & \mathbb{Z} & \times & \mathbb{N}_0 \\ & m & & n & & d \\ B = & \mathbb{Z} & \times & \mathbb{Z} & & \\ & m' & & n' & & \end{array}$$

$$Q : (m = m' \wedge n = n' \wedge m \leq n + 1)$$

$$R : (Q \wedge d = \sum_{i=m}^{n} \chi(\beta(i)))$$

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:

$$\varphi(m, n, d) = \left(d = \sum_{j=m}^{n} \chi(\beta(j)) \right),$$

az invariáns tulajdonság:

$$P = \left(Q \wedge k \in [m - 1..n] \wedge d = \sum_{j=m}^{k} \chi(\beta(j)) \right),$$

a Q' állítás:

$$Q' = (Q \wedge k = m - 1 \wedge d = 0).$$

A Q-ból Q'-be jutó program a $k, d := m - 1, 0$ értékadás, mivel
$Q \Rightarrow lf(k, d := m - 1, 0, Q') = (Q \wedge m - 1 = m - 1 \wedge 0 = 0) = Q$.

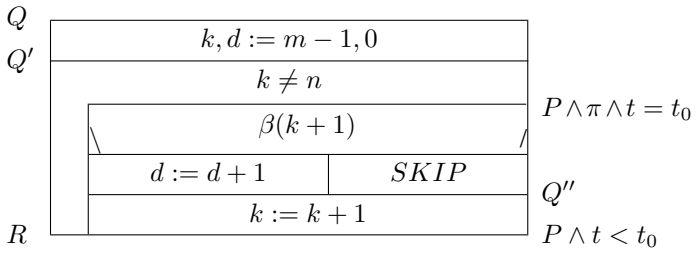
$$Q'' = \left(Q \wedge k + 1 \in [m - 1..n] \wedge d = \sum_{i=m}^{k+1} \chi(\beta(i)) \wedge t = t_0 \right)$$

Most is azt kell megvizsgálni, hogy $P \wedge \pi \wedge t = t_0$-ból következik-e Q''. Ha $\neg\beta(k + 1)$, akkor következik, míg $\beta(k + 1)$ esetén – az összegzéshez hasonlóan – meg kell növelnünk d értékét. Ezért a $P \wedge \pi \wedge t = t_0$-ból 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:

$$\begin{array}{rcl} P \wedge \pi \wedge t = t_0 \wedge \beta(k + 1) & \Rightarrow & lf(d := d + 1, Q''), \\ P \wedge \pi \wedge t = t_0 \wedge \neg\beta(k + 1) & \Rightarrow & lf(SKIP, Q'') \end{array}$$

miatt $P \wedge \pi \wedge t = t_0 \Rightarrow lf(IF, Q'')$ teljesül.
A fenti meggondolások alapján nyilvánvaló az alábbi tétel:

Tétel: Az alábbi, struktogram formában megadott program megoldása a fent specifikált feladatnak:



12.1.3. 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.

$$A = \underset{m}{\mathbb{Z}} \times \underset{n}{\mathbb{Z}} \times \underset{i}{\mathbb{Z}} \times \underset{max}{\mathcal{H}}$$
$$B = \underset{m'}{\mathbb{Z}} \times \underset{n'}{\mathbb{Z}}$$
$$Q : (m = m' \wedge n = n' \wedge m \leq n)$$
$$R : (Q \wedge i \in [m..n] \wedge max = f(i) \wedge \forall j \in [m..n] : f(j) \leq f(i))$$

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. Most

$$\varphi(m, n, i, max) = (i \in [m..n] \wedge max = f(i) \wedge \forall j \in [m..n] : f(j) \leq f(i)) .$$

Tehát az invariáns tulajdonság:

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

$$Q' = (Q \wedge k = m \wedge i = m \wedge max = f(m)),$$

és S_1 a $i, k, max := m, m, f(m)$ értékadás. A

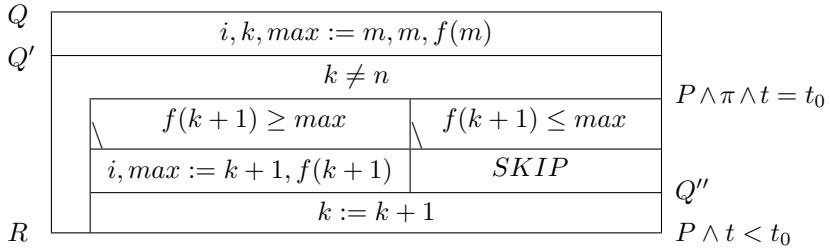
$Q'' = (Q \wedge k + 1 \in [m..n] \wedge i \in [m..k + 1] \wedge max = f(i) \wedge$
$\forall j \in [m..k + 1] : f(j) \leq f(i) \wedge t = t_0)$.

A $P \wedge \pi \wedge t = t_0$-ból Q''-be jutó program itt is egy elágazás lesz: $IF(f(k + 1) \geq max : i, max := k + 1, f(k + 1); f(k + 1) \leq max : SKIP)$, ugyanis

$$P \wedge \pi \wedge t = t_0 \wedge f(k + 1) >= max \quad \Rightarrow \quad lf(i, max := k + 1, f(k + 1), Q''),$$
$$P \wedge \pi \wedge t = t_0 \wedge f(k + 1) <= max \quad \Rightarrow \quad lf(SKIP, Q'')$$

miatt $P \wedge \pi \wedge t = t_0 \Rightarrow lf(IF, Q'')$ teljesül.

Tétel: Az alábbi, struktogram formában megadott program megoldása a fent specifikált feladatnak:



12.1.4. 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.

$$A = \underset{m}{\mathbb{Z}} \times \underset{n}{\mathbb{Z}} \times \underset{i}{\mathbb{Z}} \times \underset{max}{\mathcal{H}} \times \underset{l}{\mathbb{L}}$$

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

$$Q : (m = m' \wedge n = n' \wedge m \leq n + 1)$$

$$R : (Q \wedge l = (\exists i \in [m..n] : \beta(i)) \wedge l \to (i \in [m..n] \wedge \beta(i) \wedge max = f(i) \wedge \forall j \in [m..n] : \beta(j) \to f(j) \leq f(i)))$$

Itt újra megengedhető az üres intervallum, és ekkor az a válasz, hogy az intervallumban nincs β tulajdonságú elem.

A levezetés az előzőekhez hasonlóan:

$$P = (Q \wedge k \in [m-1..n] \wedge l = (\exists i \in [m..k] : \beta(i)) \wedge l \to (i \in [m..k] \wedge \beta(i) \wedge max = f(i) \wedge \forall j \in [m..k] : \beta(j) \to f(j) \leq f(i)))$$

$$Q' = (Q \wedge k = m - 1 \wedge l = hamis)$$

$$Q'' = (Q \wedge k + 1 \in [m-1..n] \wedge l = (\exists i \in [m..k+1] : \beta(i)) \wedge l \to (i \in [m..k+1] \wedge \beta(i) \wedge max = f(i) \wedge \forall j \in [m..k+1] : \beta(j) \to f(j) \leq f(i)))$$

$P \wedge \pi$ 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:

<table>
<tr><td colspan="4">$k, l := m-1, \downarrow$</td></tr>
<tr><td colspan="4">$k \neq n$</td></tr>
<tr><td>$\neg\beta(k+1)$</td><td>$\beta(k+1) \wedge \neg l$</td><td colspan="2">$\beta(k+1) \wedge l$</td></tr>
<tr><td rowspan="2">$SKIP$</td><td rowspan="2">$l, i, max :=$
$\uparrow, k+1, f(k+1)$</td><td>$f(k+1) \geq max$</td><td>$f(k+1) \leq max$</td></tr>
<tr><td>$i, max :=$
$k+1, f(k+1)$</td><td>$SKIP$</td></tr>
<tr><td colspan="4">$k := k+1$</td></tr>
</table>

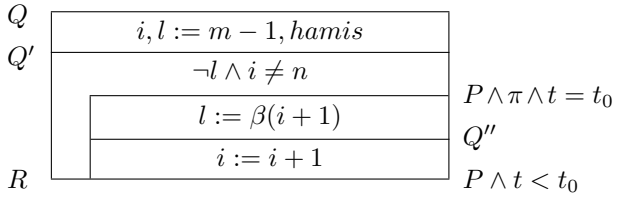
12.1.5. Lineáris keresés

Legyen $\beta : \mathbb{Z} \to \mathbb{L}$ adott függvény és $[m..n]$ egy intervallum. Keressük meg az első olyan helyet az $[m..n]$ intervallumban, ahol β igaz értéket vesz fel, ha egyáltalán van ilyen hely.

$$A = \underset{m}{\mathbb{Z}} \times \underset{n}{\mathbb{Z}} \times \underset{i}{\mathbb{Z}} \times \underset{l}{\mathbb{L}}$$
$$B = \underset{m'}{\mathbb{Z}} \times \underset{n'}{\mathbb{Z}}$$
$$Q : (m = m' \wedge n = n' \wedge m \leq n+1)$$
$$R : (Q \wedge l = (\exists j \in [m..n] : \beta(j)) \wedge l \to (i \in [m..n] \wedge \beta(i) \wedge \forall j \in [m..i-1] : \neg\beta(j)))$$

A ciklus invariáns tulajdonsága:
$P = (Q \wedge i \in [m-1..n] \wedge l = (\exists j \in [m..i] : \beta(j)) \wedge \forall j \in [m..i-1] : \neg\beta(j))$,
$Q' = (Q \wedge i = m-1 \wedge l = hamis)$,
$Q'' = (Q \wedge i+1 \in [m-1..n] \wedge l = \exists j \in [m..i+1] : \beta(j) \wedge \forall j \in [m..i] : \neg\beta(j) \wedge t = t_0)$,
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:



Megjegyzés: ezt a programozási tételt *lineáris keresés 2.8* változatnak is nevezzük.

12.2. Tételek „feltételig” változata

A továbbiakban használjuk a következő jelölést. Legyen $\delta : \mathbb{Z} \to \mathbb{L}$ és $m \in \mathbb{Z}$. Tegyük fel, hogy $\exists i \geq m : \delta(i)$. Jelöljük $\alpha(m, \delta)$-val az első olyan egész számot, ami nem

kisebb, mint m, és igaz rá a δ, azaz

$$\alpha(n,\delta) ::= min\left\{x \in \mathbb{Z} \mid x \geq m \text{ és } \delta(x)\right\}.$$

Az előző rész tételeit fogalmazzuk meg most kicsit más formában. A φ teljesülését most nem egy $[m,n]$ intervallumon, hanem m-től az első δ tulajdonságú helyig követeljük meg.

$$\begin{array}{lll} A = & \mathbb{Z} & \times \ldots \\ & m & \ldots \\ B = & \mathbb{Z} & \times \ldots \\ & m' & \ldots \end{array}$$

$Q : (m = m' \wedge \exists i \geq m : \delta(i))$

$R : (Q \wedge \varphi(m, \alpha(m,\delta)\ldots))$

A megoldás most is egy ciklus lesz. Az állapotteret egy egész és egy logikai komponenssel terjesztjük ki, legyenek a megfelelő változók k és v. A ciklus invariáns tulajdonsága pedig az, hogy $k \geq m-1$. A φ m-től k-ig teljesül, v annak megfelelően igaz vagy hamis, hogy k-ra δ igaz-e, és végül még azt is megköveteljük, hogy k előtt nincs δ tulajdonságú hely, azaz

$P = (Q \wedge k \geq m-1 \wedge \varphi(m,k,\ldots) \wedge$
$v = \exists i \in [m..k] : \delta(i) \wedge \forall j \in [m..i-1] : \neg\delta(j))$.

A ciklusfeltétel: $\neg v$, mivel $v \wedge P \Rightarrow R$.

Ebben az esetben Q'-nek választható:

$$Q' = (Q \wedge k = m-1 \wedge \varphi(m,k,\ldots) \wedge \neg v).$$

A terminálófüggvény a következő: az előfeltétel miatt létezik olyan N egész szám, ami nagyobb n-nél, és igaz rá δ; legyen

$$t = (N-k),$$

ami $P \wedge \neg v$ esetén biztosan pozitív.

Most is szekvenciaként határozzuk meg a ciklusmagot. A szekvencia második fele legyen

$$k, v := k+1, \delta(k+1)$$

és

$$Q'' = (lf(k, v := k+1, \delta(k+1), P) \wedge t = t_0) = P^{k \leftarrow k+1, v \leftarrow \delta(k+1)} \wedge t = t_0.$$

Mivel

$P^{k \leftarrow k+1, v \leftarrow \delta(k+1)} = (Q \wedge k+1 \geq m-1 \wedge \varphi(m, k+1, \ldots) \wedge$
$\delta(k+1) = \exists i \in [m..k+1] : \delta(i) \wedge \forall j \in [m..i-1] : \neg\delta(j))$,

ha $P \wedge \pi \wedge t = t_0$ teljesül, akkor Q'' teljesüléséhez már csak az kell, hogy $\varphi(m, k+1, \ldots)$ is igaz legyen.

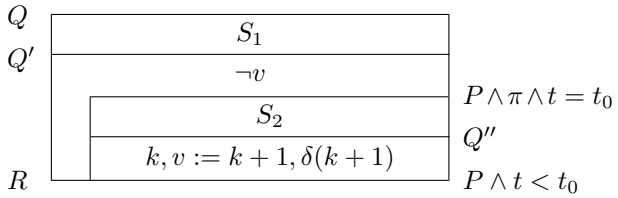
Tehát, kibővítve a paraméterteret is egy k' egész és egy v' logikai komponenssel, már csak a következő két programot keressük:

$$Q \Rightarrow lf(S_1, Q')$$

és

$$P \wedge \pi \wedge k = k' \wedge v = v' \Rightarrow lf(S_2, \varphi(m, k+1, \ldots) \wedge k = k' \wedge v = v').$$

Tehát a megoldóprogram sémája:



12.2.1. Összegzés feltételig

Legyenek adottak az $f : \mathbb{Z} \to \mathbb{Z}$ és $\delta : \mathbb{Z} \to \mathbb{L}$ függvények. Tegyük fel, hogy létezik $i \geq m$, amire δ igaz. Feladatunk az, hogy az adott m-től az első olyan i-ig, amelyre δ igaz, összegezzük az f függvény értékeit. Specifikáljuk először a feladatot.

$$A = \underset{m}{\mathbb{Z}} \times \underset{s}{\mathbb{Z}}$$
$$B = \underset{m'}{\mathbb{Z}}$$
$$Q : (m = m' \wedge \exists j \geq m : \delta(j))$$
$$R : (Q \wedge s = \sum_{j=m}^{\alpha(m,\delta)} f(j))$$

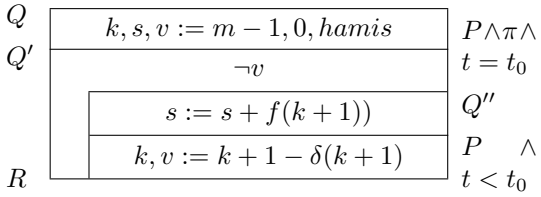
Ebben az esetben tehát

$$\varphi(m, n, s) = \left(s = \sum_{j=m}^{n} f(j) \right),$$

vagyis ugyanaz mint az intervallumos összegzésnél volt, ezért S_1-nek választhatjuk a $k, s, v := m - 1, 0, hamis$ értékadást és S_2-nek ugyanazt, mint amit az intervallumos összegzésnél: $s := s + f(k + 1)$.

Tehát kimondható a következő tétel:

Tétel: Az alábbi, struktogram formában megadott program megoldása a fent specifikált feladatnak:



12.2.2. Számlálás feltételig

Legyenek β és δ az egész számokon értelmezett logikai függvények. Tegyük fel, hogy létezik $i \geq m$, amelyre δ igaz. A feladat az, hogy számoljuk meg, hány helyen igaz β az m-től az első olyan i-ig, amire δ igaz.

$$A = \underset{m}{\mathbb{Z}} \times \underset{d}{\mathbb{N}_0}$$

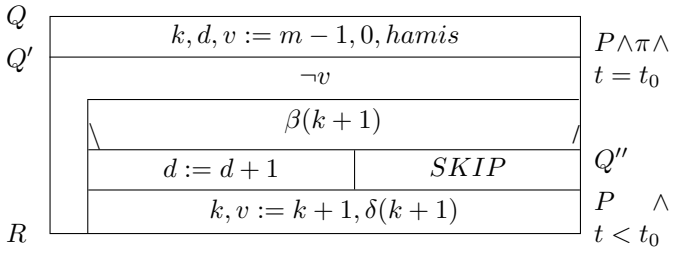
$B = \underset{m'}{\mathbb{Z}}$

$Q : (m = m' \wedge \exists i \geq m : \delta(i))$

$R : (Q \wedge d = \sum_{i=m}^{\alpha(m,\delta)} \chi(\beta(i)))$

A feladat megoldása analóg az összegzés tételénél leírtakkal.

Tétel: Az alábbi, struktogram formában megadott program megoldása a fent specifikált feladatnak:



12.2.3. Maximumkeresés feltételig

Legyen $\mathcal{H}$ egy tetszőleges rendezett halmaz és $f : \mathbb{Z} \to \mathcal{H}$ egy adott függvény és δ az egész számokon értelmezett logikai függvény. Tegyük fel, hogy létezik $i \geq m$, amire δ igaz. Feladatunk az, hogy az m-től az első olyan i-ig, amelyre δ igaz, keressük meg az f függvény maximumát és egy olyan helyét, ahol ezt a maximumértéket felveszi.

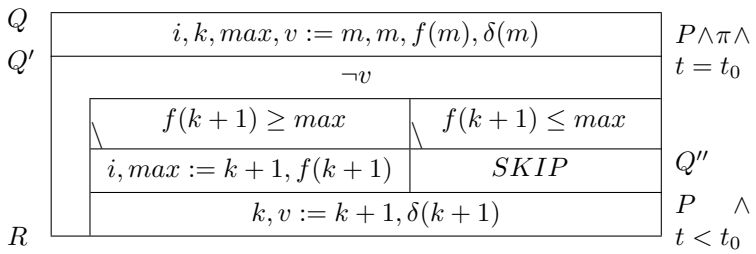
$A = \underset{m}{\mathbb{Z}} \times \underset{i}{\mathbb{Z}} \times \underset{max}{\mathcal{H}}$

$B = \underset{m'}{\mathbb{Z}}$

$Q : (m = m' \wedge \exists i \geq m : \delta(i) mbox)$

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

Tétel: Az alábbi, struktogram formában megadott program megoldása a fent specifikált feladatnak:



12.2.4. Feltételes maximumkeresés feltételig

Legyen $\mathcal{H}$ egy tetszőleges rendezett halmaz és $f : \mathbb{Z} \to \mathcal{H}$ egy adott függvény. Legyenek β és δ az egész számokon értelmezett logikai függvények. Határozzuk meg a $\lceil \beta \rceil \cap [m..\alpha(m,\delta)]$ halmaz felett, ha az nem üres, az f függvény maximumát és a halmaz egy olyan elemét, amelyen f a maximumértékét felveszi.

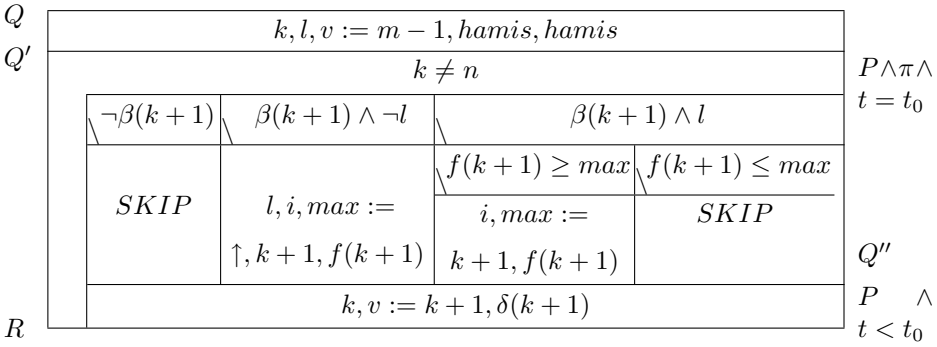
$$A = \underset{m}{\mathbb{Z}} \times \underset{i}{\mathbb{Z}} \times \underset{max}{\mathcal{H}} \times \underset{l}{\mathbb{L}}$$

$$B = \underset{m'}{\mathbb{Z}}$$

$Q : (m = m' \wedge \exists j \geq m : \delta(j))$

$R : (Q \wedge l = (\exists i \in [m..\alpha(m,\delta)] : \beta(i)) \wedge l \rightarrow (i \in [m..\alpha(m,\delta)] \wedge \beta(i) \wedge$
$max = f(i) \wedge \forall j \in [m..\alpha(m,\delta)] : \beta(j) \rightarrow (f(j) \leq f(i))))$

Tétel: Az alábbi, struktogram formában megadott program megoldása a fent specifikált feladatnak:



12.2.5. Lineáris keresés

Most egy kicsit eltérünk az általános sémától, és több speciális esetet vizsgálunk meg. Legyen $\beta : \mathbb{Z} \rightarrow \mathbb{L}$ adott tulajdonság. Az első feladat az, hogy keressük meg azt a legkisebb β tulajdonságú egész számot, amely nem kisebb, mint az adott $m \in \mathbb{Z}$ szám, feltéve, hogy van ilyen.

$$A = \underset{m}{\mathbb{Z}} \times \underset{i}{\mathbb{Z}}$$

$$B = \underset{m'}{\mathbb{Z}}$$

$Q : (m = m' \wedge \exists j \geq m : \beta(j))$

$R : (Q \wedge i \geq m \wedge \beta(i) \wedge \forall j \in [m..i-1] : \neg\beta(j))$

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) $Q' = (Q \wedge i = m)$

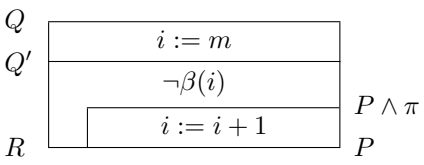
2) $\pi = \neg\beta(i)$

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:



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 $t = N - i$.

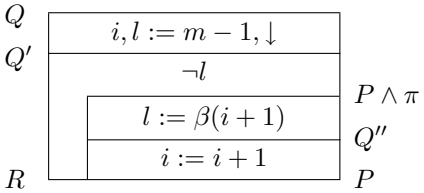
5) Az $i := i + 1$ értékadás csökkenti a terminálófüggvény értékét.

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.



Legyenek adottak a $\gamma, \delta : \mathbb{Z} \to \mathbb{L}$ és az m egész szám. Jelöljük β-val a $\gamma \vee \delta$-t, és tegyük fel, hogy létezik $j \geq m$, hogy $\beta(j)$. 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.

$$A = \underset{m}{\mathbb{Z}} \times \underset{i}{\mathbb{Z}} \times \underset{u}{\mathbb{L}}$$

$$B = \underset{m'}{\mathbb{Z}}$$

$Q : (m = m' \wedge \exists j \geq m : \beta(j))$

$R : (Q \wedge u = (\exists j \geq m : \gamma(j) \wedge \forall k \in [m..j-1] : \neg\delta(k)) \wedge$
$u \to (i \geq m \wedge \gamma(i) \wedge \forall j \in [m..i-1] : \neg\beta(j)))$

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 $t = N - i$.

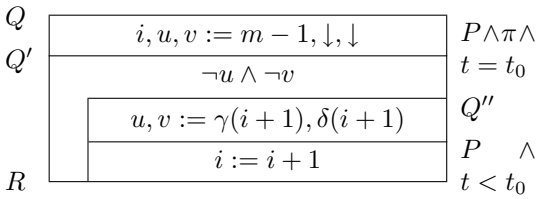
5) Az $i := i + 1$ értékadás csökkenti a terminálófüggvény értékét.

4) A ciklusmag szekvencia lesz, melynek közbülső feltétele ($lf(i := i + 1, P)$):

$$Q'' = (Q \wedge i + 1 \geq m - 1 \wedge u = (\exists j \in [m..i+1] : \gamma(j)) \wedge \\ v = (\exists j \in [m..i+1] : \delta(j)) \wedge \forall j \in [m..i] : \neg\beta(j) \wedge t = t_0),$$

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:



Megjegyzés: A lineáris keresés fenti három változatát rendre a *lineáris keresés 1., 2., 3.* változatának is szoktuk nevezni.

12.2.6. Tételek másképpen

A fejezetben szereplő tételeket kimondhattuk volna kicsit más formában, formákban is. Ezzel a tételek különféle változatat kaphatjuk. Nézzünk néhány példát!

Eddig a „feltételes” tételeinket úgy fogalmaztuk meg, hogy φ teljesülését m-től az első δ tulajdonságú helyig követeljük meg, ezt a δ tulajdonságú helyet is beleértve. Kimondhatnánk őket úgy is, hogy φ teljesülését m-től csak a $\neg\delta$ tulajdonságú helyekre követeljük meg.

Tegyük fel, hogy $\exists i \geq m - 1 : \delta(i)$, és legyen

$$\alpha'(m, \delta) ::= max\left\{x - 1 \in \mathbb{Z} \mid x \geq m \text{ és } \neg\delta(x)\right\}.$$

$A = \begin{matrix}\mathbb{Z} & \times & \dots \\ m & & \dots\end{matrix}$

$B = \begin{matrix}\mathbb{Z} & \times & \dots \\ m' & & \dots\end{matrix}$

$Q : (m = m' \wedge \exists i \geq m : \delta(i))$

$R : (Q \wedge \varphi(m, \alpha'(m, \delta) \dots))$

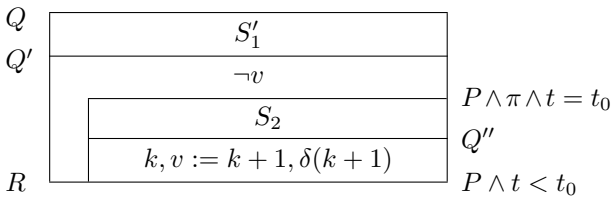
A megoldás nagyon hasonló lesz az előzőhöz, csak az invariáns tulajdonságban $[m..k]$ helyett $[m..k+1]$ szerepel

$P_1 = (Q \wedge k \geq m - 1 \wedge \varphi(m, k, \dots) \wedge$
$v = \exists i \in [m..k+1] : \delta(i) \wedge \forall j \in [m..i-1] : \neg\delta(j))$

és Q'-ben pedig $\neg v$ helyett $v = \delta(k)$

$$Q_1' = (Q \wedge k = m - 1 \wedge \varphi(m, k, \dots) \wedge v = \delta(k)) .$$

Innen kezdve a levezetés teljesen azonos módon megy, és a megoldóprogram sémája:



ahol S_1' specifikációja

$$Q \Rightarrow lf(S_1', Q_1'),$$

ami azt jelenti, hogy a konkrét tételekbe $\dots, v := \dots, hamis$ helyett $\dots, v := \dots, \delta(m)$ kerül.

Egy másik lehetőség változatokra: az ebben és az előző részben szereplő programozási tételeket megfogalmazhattuk volna általánosabban is.

A függvényeket értelmezhetjük az egészek helyett egy tetszőleges, olyan halmazon, amelynek minden elemének van rákövetkezője $(succ)$ és megelőzője $(pred)$. Ez semmi különös problémát nem okoz, csak $x + 1$ helyett $succ(x)$-et, $x - 1$ helyett $pred(x)$-et kell írni.

Kicsit bonyolultabb a helyzet, ha csak azt tesszük fel, hogy a halmaz bármelyik eleméből megkaphatjuk bármelyik elemét a $(succ)$ vagy a $(pred)$ véges sokszori alkalmazásával, ami egyébként sokszor előforduló eset a gyakorlatban. A megoldás ebben az esetben hasonló, a P és a Q' módosításával új S_1' programot specifikálunk.

12.3. Bonyolultabb feladatok

Ebben a részben néhány kevésbé egyszerű tételt vezetünk le elsősorban azért, hogy megmutassuk, összetettebb feladatok esetén is használható a levezetés, másrészt az így kapott tételek is elég fontosak.

12.3.1. 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, 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.

$$\begin{array}{lccccccc} A = & \mathbb{Z} & \times & \mathbb{Z} & \times & \mathcal{H} & \times & \mathbb{Z} & \times & \mathbb{L} \\ & m & & n & & h & & i & & l \end{array}$$

$$\begin{array}{lccccc} B = & \mathbb{Z} & \times & \mathbb{Z} & \times & \mathcal{H} \\ & m' & & n' & & h' \end{array}$$

$$Q : (m = m' \wedge n = n' \wedge h = h' \wedge m \leq n + 1 \wedge \forall k, j \in [m..n] : (k < j) \to (f(k) \leq f(j)))$$

$$R : (Q \wedge l = (\exists j \in [m..n] : f(j) = h) \wedge l \to (i \in [m..n] \wedge f(i) = h))$$

A monotonitást felhasználva az intervallumot mindkét végéről szűkítjük az invariánsban:

$$P = (Q \wedge [u..v] \subseteq [m..n] \wedge \forall j \in [m..n] \setminus [u..v] : f(j) \neq h \wedge \\ l \rightarrow (i \in [u..v] \wedge f(i) = h))$$

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:

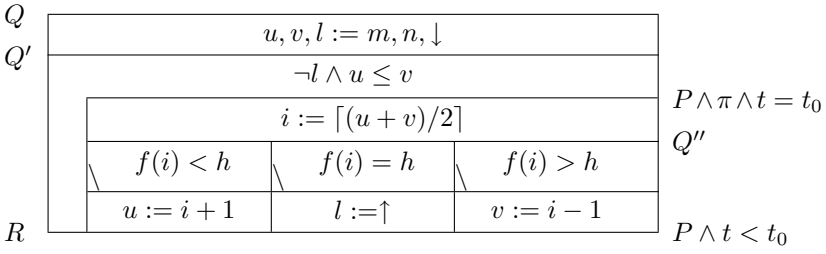
$$Q'' = (Q \wedge [u..v] \subseteq [m..n] \wedge \forall j \in [m..n] \setminus [u..v] : f(j) \neq h \wedge \\ \neg l \wedge (i \in [u..v]))$$

Ekkor a szekvencia első fele lehetne az $i :\in [u..v]$ értékkiválasztás. Hatékonysági szempontokat is figyelembe 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 a $v := i - 1$ értékadás az invariánst megtartja.

5) Egyszerűen ellenőrizhető, hogy a fenti elágazás mindhárom ága csökkenti a terminálófüggvényt.

Tétel: Az alábbi, struktogram formában megadott program megoldása a fent specifikált feladatnak:



12.3.2. Visszalépéses keresés

Legyen $N \in \mathbb{N}$, és $N > 1$. Legyenek U_i ($i \in [1..N]$) tetszőleges véges, legalább kételemű halmazok ($1 < \sigma_i = |U_i| < \infty$). $U = U_1 \times \cdots \times U_N$.

Legyen $\varrho : U \rightarrow \mathbb{L}$, amely felbontható $\varrho_i : U \rightarrow \mathbb{L}$ ($i \in [0..N]$) tulajdonságok sorozatára az alábbi módon:

1. $\varrho_0 = igaz$;

2. $\forall i \in [0..N-1] : \forall u \in U : \varrho_{i+1}(u) \rightarrow \varrho_i(u)$;

3. $\forall i \in [1..N] : \forall u, v \in U : (\forall j \in [1..i] : u_j = v_j) \rightarrow \varrho_i(u) = \varrho_i(v)$;

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.

$$
\begin{aligned}
&A = \underset{u}{U} \times \underset{l}{\mathbb{L}} \\
&B = \{\mathcal{X}\} \\
&Q: \ Igaz \\
&R : (l = \exists v \in U : \varrho(v) \wedge l \rightarrow (u \in U \wedge \varrho(u)))
\end{aligned}
$$

Számozzuk meg U elemeit a következő 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 $0 \leq i_1 < \sigma_1, \ldots, 0 \leq i_N < \sigma_N$. E megszámozás egy lexikografikus rendezést definiál 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 $\mathcal{N} \rightarrow U$ leképezést φ.

Vegyük észre, hogy az $\mathcal{N}$ halmaz elemei felfoghatók vegyes alapú számrendszerben felírt számként is. Ez alapján egy $\nu \in \mathcal{N}$ N-es számértéke:

$$
\begin{aligned}
f(\nu) &= \sum_{i=1}^{N} \nu_i \cdot \Omega_i, \qquad \text{ahol:} \\
\Omega_i &= \prod_{j=i+1}^{N} \sigma_j \quad (i \in [1..N]).
\end{aligned}
$$

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:

$$
\begin{aligned}
&A = \underset{\nu}{\mathcal{N}} \times \underset{l}{\mathbb{L}} \\
&B = \{\mathcal{X}\} \\
&Q: \ Igaz \\
&R : (l = \exists \mu \in \mathcal{N} : \varrho(\varphi(\mu)) \wedge \\
&\qquad l \rightarrow (\varrho(\varphi(\nu)) \wedge \forall \mu \in \mathcal{N} : f(\mu) < f(\nu) \rightarrow \neg\varrho(\varphi(\mu))))
\end{aligned}
$$

Ha nem használjuk ki a ϱ speciális tulajdonságait, akkor a fenti feladat megoldható lineáris kereséssel, a $[0..|\mathcal{N}| - 1]$ intervallumon. Vizsgáljuk meg, hogyan használhatnánk ki ϱ specialitását! Nevezetesen azt, hogy a ϱ 3. tulajdonsága miatt, ha $\varrho_i(\varphi(\nu))$ igaz, $\varrho_{i+1}(\varphi(\nu))$ 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.

Legyen $\varepsilon_0 = (0, \ldots, 0) \in \mathcal{N}$ és $\forall i \in [1..N] : \varepsilon_i \in \mathcal{N}$ olyan, hogy $\forall j \in [1..N] \setminus \{i\} : \varepsilon_{i_j} = 0$ és $\varepsilon_{i_i} = 1$. Nyilvánvaló, hogy $f(\varepsilon_0) = 0$, $f(\varepsilon_N) = 1$ és $\forall j \in [1..N] : f(\varepsilon_i) = \Omega_i$.

Egészítsük ki a ν-t egy „túlcsordulásbittel”, amelynek 1 értéke azt jelzi, hogy ν értéke már nem növelhető.

Terjesszük ki az f függvényt az alábbi módon:

$f : \{0,1\} \times \mathcal{N} \to \mathbb{N}_0$,

$$f(\nu_0, \nu) = \nu_0 * \Omega_0 + \sum_{i=1}^{n} \nu_i \cdot \Omega_i, \qquad \text{ahol:}$$

$$\Omega_0 = \prod_{j=1}^{N} \sigma_j.$$

Ezeket a jelöléseket használva definiáljuk az összeadás műveletét két $\mathcal{N}$-beli elem között.

$$\nu' + \nu'' = \nu''' \iff f(\nu') + f(\nu'') = f(\nu_0, \nu''')$$

Most már megfogalmazhatunk egy egyszerű, de fontos állítást, ami alapján kihasználhatjuk a megoldás során ϱ speciális voltát.

12.1. állítás: *Legyen $\nu \in \mathcal{N}$, valamilyen $i \in [1..N]$-re $\neg\varrho_i(\varphi(\nu))$, és $\forall j \in [i+1..N]$: $\nu_j = 0$. Ekkor $\forall\mu$-re, ami nagyobb, mint ν, és kisebb, mint $\nu + \varepsilon_i$, $\neg\varrho_i(\varphi(\nu))$.*

A következőkben a vegyes alapú számrendszerbeli számot típusnak tekintve definiálunk két típusműveletet is, a fenti állítást is figyelembe véve.

Az első művelet ν megnövelése ε_m-mel.

$$\begin{array}{lccccc} A_{növel} = & \mathcal{N} & \times & \mathbb{N}_0 & \times & \{0,1\} \\ & \nu & & m & & \nu_0 \end{array}$$

$$\begin{array}{lccc} B_{növel} = & \mathcal{N} & \times & \mathbb{N}_0 \\ & \nu' & & m' \end{array}$$

$$Q_{növel} : (\nu = \nu' \wedge m = m' \wedge m' \in [1..N] \wedge \forall i \in [m'+1..N] : \nu_i = 0)$$

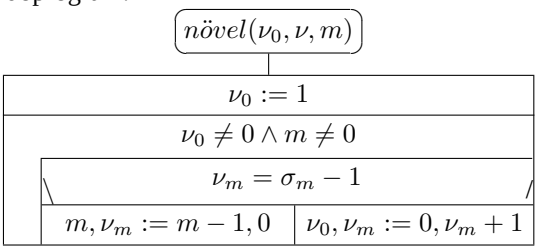
$$R_{növel} : ((\nu_0, \nu) = \nu' + \varepsilon_{m'} \wedge m \in [0..m'] \wedge \forall i \in [m+1..N] : \nu_i = 0 \wedge \nu_m \neq 0)$$

A megoldás egy ciklus, amelynek invariáns tulajdonsága:

$$P_{növel} : (f(\nu) + \nu_0 * Q_m = f(\nu') + Q_{m'} \wedge m \in [0..m'] \wedge \forall i \in [m+1..N] : \nu_i = 0 \wedge \forall i \in [1..m-1] : \nu_i = \nu'_i)$$

Ciklusfeltétel: $\nu_0 \neq 0 \wedge m \neq 0$, és a terminálófüggvény: $\nu_0 + m$.

Ekkor a megoldóprogram:



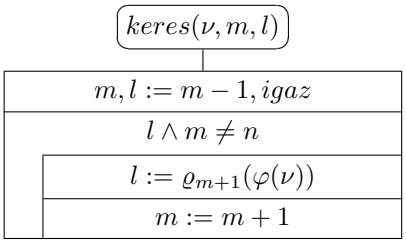
Definiáljuk a másik műveletet is, amely amellett, hogy $\varrho(\varphi(\nu))$-t eldönti, azt a legkisebb indexet is megadja, amelyre $\varrho_i(\varphi(\nu))$ hamis.

$$\begin{array}{lccccc} A_{keres} = & \mathcal{N} & \times & \mathbb{N}_0 & \times & \mathbb{L} \\ & \nu & & m & & l \end{array}$$

$$\begin{array}{lccc} B_{keres} = & \mathcal{N} & \times & \mathbb{N}_0 \\ & \nu' & & m \end{array}$$

$$Q_{keres} : (\nu = \nu' \wedge m = m' \wedge m' \in [1..N] \wedge \varrho_{m'-1}(\varphi(\nu)))$$
$$R_{keres} : (\nu = \nu' \wedge l = \varrho(\varphi(\nu)) \wedge$$
$$\neg l \rightarrow (m \in [m'..N] \wedge \varrho_{m-1}(\varphi(\nu)) \wedge \neg\varrho_m(\varphi(\nu)) \wedge$$
$$l \rightarrow m = N))$$

Ennek a feladatnak a levezetése majdnem azonos a lineáris keresésével.



Megjegyezzük, hogy az utófeltétel utolsó sora esetünkben fölösleges, de a visszalépéses számlálásnál majd építünk rá.

Mindezek birtokában már egyszerű lesz a megoldóprogram levezetése. Legyen az invariáns tulajdonság:

$$P = (\forall \mu \in \mathcal{N} : 0 \leq f(0,\mu) < f(\nu_0,\nu) \rightarrow \neg\varrho(\varphi(\mu)) \wedge l = \varrho(\varphi(\nu)) \wedge$$
$$\neg l \rightarrow (\nu_0 = 1 \vee m \in [1..N] \wedge \neg\varrho_m(\varphi(\nu)) \wedge \varrho_{m-1}(\varphi(\nu)) \wedge$$
$$\forall i \in [m+1..N] : \nu_i = 0))$$

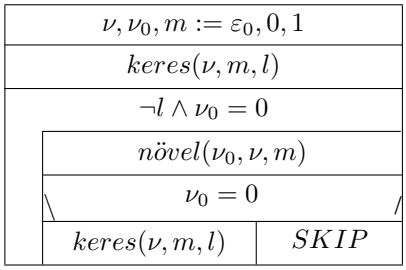
A ciklusfeltétel és a terminálófüggvény kézenfekvő módon adódik: $\neg l \wedge \nu_0 = 0$ és $\prod_{j=1}^{N} \sigma_j - f(\nu_0,\nu)$.

Az invariáns tulajdonság teljesülését a
$Q' = (\nu = \varepsilon_0 \wedge \nu_0 = 0 \wedge$
$l = \varrho(\varphi(\varepsilon_0)) \wedge \neg l \rightarrow (m \in [1..N] \wedge \neg\varrho_m(\varphi(\nu)) \wedge \varrho_{m-1}(\varphi(\nu))))$
garantálja.

Q'-t egy szekvenciával érjük el, amelynek a közbülső tulajdonsága
$Q'' = (\nu = \varepsilon_0 \wedge \nu_0 = 0 \wedge m = 1)$.
A Q''-ből következik Q_{keres}, ezért a szekvencia második tagjának a $keres(\nu,m,l)$-t választva teljesül R_{keres}, amiből pedig következik Q'. Természetesen a szekvencia első tagjának a $\nu, \nu_0, m := \varepsilon_0, 0, 1$ értékadást választjuk.

A ciklusmag levezetése is egyszerű. $P \wedge \pi$-ből következik $Q_{növel}$, ezért a ciklusmag első felének $növel(\nu_0,\nu,m)$-t választva igaz lesz $R_{növel}$, sőt P-t és a 12.1. állítást figyelembe véve az is igaz, hogy $\forall \mu \in \mathcal{N} : 0 \leq f(0,\mu) < f(\nu_0,\nu) \rightarrow \neg\varrho(\varphi(\mu))$. Abban az esetben, ha $\nu_0 = 1$ is igaz, P is teljesül. Ha $\nu_0 = 1$, akkor még
$\neg l = \varrho(\varphi(\nu)) \wedge$
$\neg l \rightarrow (\nu_0 = 1 \vee m \in [1..N] \wedge \neg\varrho_m(\varphi(\nu)) \wedge \varrho_{m-1}(\varphi(\nu)) \wedge$
$\forall i \in [m+1..N] : \nu_i = 0)$
teljesüléséről kell gondoskodni, de ehhez R_{keres} már elég lenne, az pedig igaz lesz $keres(\nu,m,l)$ után, mert Q_{keres} igaz volt.

Mivel $növel(\nu_0,\nu,m)$ nyilván csökkenti a terminálófüggvény értékét, a következő program megoldása a feladatnak:



12.3.3. Visszalépéses számlálás

A visszalépéses kereséshez hasonlóan több visszalépéses technikát alkalmazó algoritmus is levezethető, például a visszalépéses számlálás.

$A = \underset{\nu}{\mathcal{N}} \times \underset{d}{\mathbb{N}_0}$

$B = \{\mathcal{X}\}$

$Q : Igaz$

$R : \left(d = \sum_{\mu < (\nu_0, \nu)} \varrho(\varphi(\mu))\right)$

A feladat megoldása most is egy ciklus lesz, amelynek az invariáns tulajdonsága

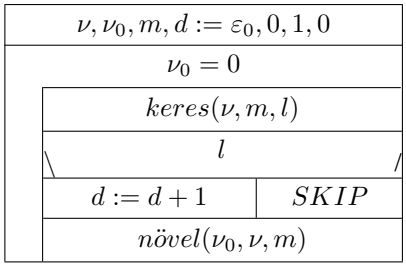
$$P = (d = \sum_{\mu < (\nu_0, \nu)} \varrho(\varphi(\mu)) \wedge (\nu_0 = 1 \vee m \in [1..N] \wedge \varrho_{m-1}(\varphi(\nu)) \wedge \forall i \in [m+1..N] : \nu_i = 0)).$$

Az invariáns második sora garantálja a 12.1. állítás alkalmazhatóságát. Most az invariáns teljesülését egy szimultán értékadással érhetjük el: $\nu, c, m, d := \varepsilon_0, 0, 1, 0$.

A ciklusfeltétel $\nu_0 = 0$ lesz. A terminálófüggvény ugyanaz, mint a visszalépéses keresésnél.

Az invariáns tulajdonságból következik Q_{keres}, ezért $keres(\nu, m, l)$ után R_{keres} igaz lesz. Ha $\neg l$, akkor $\left(\sum_{\mu < (\nu_0, \nu)} \varrho(\varphi(\mu)) = \sum_{\mu \leq (\nu_0, \nu)} \varrho(\varphi(\mu))\right)$, egyébként $\left(\sum_{\mu < (\nu_0, \nu)} \varrho(\varphi(\mu)) + 1 = \sum_{\mu \leq (\nu_0, \nu)} \varrho(\varphi(\mu))\right)$.

Végül, a $növel(\nu_0, \nu, m)$ megtartja az invariáns tulajdonságot a 12.1. állítás miatt, és csökkenti a terminálófüggvény értékét. Megjegyezzük, hogy itt használtuk ki azt, hogy a keresési feladatban $\varrho(\varphi(\nu)))$ esetén is meghatároztuk, mi legyen m értéke.



12.4. 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:

$$\begin{array}{lll} A = & X & \times\ Y \\ & x & \quad y \end{array}$$

$$\begin{array}{lll} B = & X & \\ & x' & \end{array}$$

$Q : (x = x')$

$R : (y = f(x'))$

Természetesen az $y := f(x)$ értékadás megoldása a feladatnak. Ha ez az értékadás nem megengedett program, és semmi mást nem tudunk a függvényről, akkor a megoldóprogram előállításáról sem tudunk mondani semmit. Ezért az elkövetkezőkben további feltételezésekkel fogunk élni.

12.4.1. Függvénykompozícióval adott függvény kiszámítása

Tegyük fel, hogy $f = h \circ g$, ahol $g : X \to Z$ és $h : Z \to Y$ függvények.

Ekkor a feladat megoldása egy szekvencia lesz. Kibővítjük az állapotteret egy újabb (Z típusú) komponenssel, melynek változója legyen z. A szekvencia közbülső feltétele legyen

$Q' : (z = g(x'))$.

Tétel: A kompozíció helyettesítési értékét kiszámító program:

$z := g(x)$
$y := h(z)$

12.4.2. 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:

$$f(x) = \begin{cases} g_1(x), & \text{ha } \pi_1(x); \\ g_2(x), & \text{ha } \pi_2(x); \\ \vdots & \vdots \\ g_n(x), & \text{ha } \pi_n(x). \end{cases}$$

Az elágazás első feltétele f definíciója miatt teljesül. Az $y := g_i(x)$ értékadások garantálják a második teljesülését is.

Tétel: Az esetszétválasztással adott függvény értékét kiszámoló program:

$\pi_1(x)$	$\pi_2(x)$	$\ldots$	$\pi_n(x)$
$y := g_1(x)$	$y := g_2(x)$	$\ldots$	$y := g_n(x)$

12.4.3. 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 $f : \mathbb{Z} \to H$ parciális függvényt az alábbi módon:

$$\begin{aligned} f(m) &= t_0, \\ f(m-1) &= t_{-1}, \\ &\vdots \quad\quad \vdots \\ f(m-k+1) &= t_{-k+1}; \end{aligned}$$

továbbá $\forall i \geq m$:

$$f(i+1) = F(i+1, f(i), ..., f(i-k+1)).$$

Feladatunk az, hogy meghatározzuk az f függvény $n \geq m$ helyen felvett értékét.

$$\begin{array}{lccccccc} A = & \mathbb{Z} & \times & \mathbb{Z} & \times & H & \times \ldots \times & H \\ & m & & n & & y & t_0 \quad \ldots & t_{-k+1} \\ B = & \mathbb{Z} & \times & \mathbb{Z} & \times & H & \times \ldots \times & H \\ & m' & & n' & & t'_0 & \ldots & t'_{-k+1} \end{array}$$

$Q : (m = m' \wedge n = n' \wedge n \geq m \wedge t_0 = t'_0 \wedge \cdots \wedge t_{-k+1} = t' - k + 1)$

$R : (Q \wedge y = f(n))$

Ez a feladat is a fejezet elején tárgyalt intervallumos feladatok közé tartozik. Levezetése ennek megfelelően ugyanúgy történik. A ciklus invariáns tulajdonsága, a Q' és a Q'':

$P = (Q \wedge i \in [m..n] \wedge y = f(i), y_{-1} = f(i-1) \wedge \cdots \wedge y_{-k+1} = f(i-k+1)),$

$Q' = (Q \wedge i = m \wedge y = t_0 \wedge y_{-1} = t_{-1} \wedge \cdots \wedge y_{-k+1} = t_{-k+1}),$

$Q'' = (Q \wedge i+1 \in [m..n] \wedge y = f(i+1) \wedge y_{-1} = f(i) \wedge \ldots$
$\wedge\, y_{-k+1} = f(i-k+2)).$

Az $i, y, y_{-1}, .., y_{-k+1} := 0, t_0, t_{-1}, .., t_{-k+1}$ szimultán értékadás Q-ból Q'-be jut, és $P \wedge \pi$-ből következik

$lf(y, y_{-1}, .., y_{-k+1} := F(i+1, y, ..., y_{-k+1}), y, .., y_{-k+2}), Q'').$

Variáns függvénynek $n - i$-t választva az $i := i + 1$ értékadás csökkenti azt.

Tétel: Az alábbi struktogrammal adott program megoldása a specifikált feladatnak:

$i, y, y_{-1}, .., y_{-k+1} := m, t_0, t_{-1}, .., t_{-k+1}$	
$i \neq n$	
	$y, y_{-1}, .., y_{-k+1} := F(i+1, y, ..., y_{-k+1}), y, .., y_{-k+2})$
	$i := i + 1$

Megjegyezzük, hogy a gyakorlatban k nagyon sokszor egyenlő eggyel, ezért erre az esetre külön is felírjuk a megoldóprogramot.

$i, y := m, t_0$	
$i \neq n$	
	$y := F(i+1, y)$
	$i := i + 1$

12.4.4. 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 $\forall i \in [1..n] : X_i = \{x \in \wp(H_1) : |x| < \infty\}$, azaz $\mathfrak{F}(H_1)$ és $\forall i \in [1..m] : Y_i = \{y \in \wp(H_2) : |y| < \infty\}$, azaz $\mathfrak{F}(H_2)$. 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.

12.1. 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.

12.2. 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, $X_1 = X_2 = Y = \{x \in \wp(H) : |x| < \infty\}$, $f : X_1 \times X_2 \to Y$, $f((x_1, x_2)) = x_1 \cup x_2$. Ekkor f elemenként feldolgozható, ugyanis 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 függvény.

A következő tételt gyakran fogjuk használni arra, hogy elemenként feldolgozható függvényt definiáljunk.

12.1. TÉTEL: ELÉGSÉGES FELTÉTEL ELEMENKÉNTI FELDOLGOZHATÓSÁGRA
Legyen $X = X_1 \times ... \times X_n, Y = Y_1 \times ... \times Y_m, X_i = \mathfrak{F}(H_1) \quad i \in [1..n]$ és $Y_i = \mathfrak{F}(H_2) \quad i \in [1..m]$.

Az $f : X \to Y$ függvény elemenként feldolgozható ha

$$\forall j \in [1..m] : f_j(x_1, \ldots, x_n) = \bigcup_{e \in x_1 \cup \cdots \cup x_n} f_j(s_1(e), \ldots, s_n(e))$$

és

$$\forall a, b \in x_1 \cup \cdots \cup x_n, a \neq b :$$

$$\forall j \in [1..m] : f_j(s_1(a), \ldots, s_n(a)) \cap f_j(s_1(b), \ldots, s_n(b)) = \emptyset,$$

ahol

$$\forall i \in [1..n] : s_i(e) = \begin{cases} \emptyset & \text{ha } e \notin x_i, \\ \{e\} & \text{ha } e \in x_i. \end{cases}$$

Bizonyítás: A tétel egyszerűen következik abból, hogy ha $\overline{x}, \overline{\overline{x}}$ teljesen diszjunkt felbontása x-nek, akkor

$$(\overline{x_1} \cup \cdots \cup \overline{x_n}) \cup (\overline{\overline{x_1}} \cup \cdots \cup \overline{\overline{x_n}}) = x_1 \cup \cdots \cup x_n$$

és

$$(\overline{x_1} \cup \cdots \cup \overline{x_n}) \cap (\overline{\overline{x_1}} \cup \cdots \cup \overline{\overline{x_n}}) = \emptyset.$$

□

A továbbiakban az elemenként feldolgozható függvények helyettesítési értékének kiszámításával fogunk foglalkozni.

Mielőtt belekezdenénk a feladat specifikálásába és megoldásába, bevezetünk két olyan, halmazokra vonatkozó parciális értékadást, amelyeket aztán a megoldóprogramokban primitív műveletnek tekintünk.

- Legyen H egy tetszőleges halmaz, és definiáljuk az $f_{\widetilde{\cup}} :\in \mathfrak{F}(H) \times H \to \mathfrak{F}(H)$ parciális függvényt:
$$f_{\widetilde{\cup}}(h, e) = h \cup \{e\}, \text{ha } e \notin h.$$

- Ugyanezt a jelölést használjuk akkor is, ha $f_{\widetilde{\cup}} :\in \mathfrak{F}(H) \times \mathfrak{F}(H) \to \mathfrak{F}(H)$ és
$$f_{\widetilde{\cup}}(h, g) = h \cup g, \text{ ha } h \cap g = \emptyset.$$

- Hasonlóan, legyen H egy tetszőleges halmaz, és definiáljuk az $f_{\simeq} :\in \mathfrak{F}(H) \times H \to \mathfrak{F}(H)$ parciális függvényt:
$$f_{\simeq}(h, e) = h \setminus \{e\}, \text{ ha } e \in h.$$

- Ebben az esetben is, ha $f_{\simeq} :\in \mathfrak{F}(H) \times \mathfrak{F}(H) \to \mathfrak{F}(H)$ parciális függvény:
$$f_{\simeq}(h, g) = h \setminus g, \text{ ha } g \subseteq h.$$

A fenti függvényeket kiszámító $h := f_{\widetilde{\cup}}(h, x)$, illetve $h := f_{\simeq}(h, x)$ parciális értékadásokat a továbbiakban $h := h \mathbin{\widetilde{\cup}} x$-szel és $h := h \simeq x$-szel fogjuk jelölni.

Egyváltozós egyértékű eset

Először vizsgáljuk meg azt az esetet, amikor mind X, mind Y egykomponensű, azaz $m = 1$ és $n = 1$. Ekkor az f függvény egy halmazhoz egy másik halmazt rendel.

$$\begin{array}{lll} A = & X & \times Y \\ & x & y \\ B = & X & \\ & x' & \end{array}$$

$Q : (x = x')$

$R : (y = f(x'))$

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 teljesül (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 terminálófüggvénynek 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:

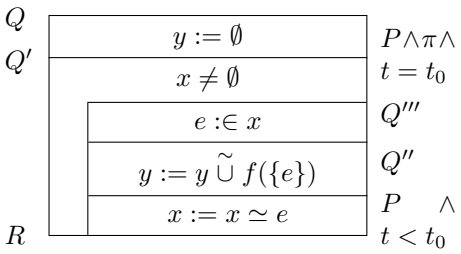
$$\begin{array}{rcl} f(\{e\}) \cup f(x \setminus \{e\}) & = & f(x) \\ f(\{e\}) \cap f(x \setminus \{e\}) & = & \emptyset \end{array}$$

   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 a következő program megoldása a specifikált feladatnak:



Bizonyítás: A tétel a fenti levezetésből következik.

Kétváltozós egyértékű eset

Legyen $f : X \times Y \to Z$ $(X, Y, Z \subseteq \mathfrak{F}(H)$ elemenként feldolgozható függvény.

$$A = \underset{x}{X} \times \underset{y}{Y} \times \underset{z}{Z}$$
$$B = \underset{x'}{X} \times \underset{y'}{Y}$$
$$Q : (x = x' \wedge y = y')$$
$$R : (z = f(x', y'))$$

Tétel: Ekkor a következő program megoldása a specifikált feladatnak:

$z := \emptyset$		
$x \neq \emptyset \vee y \neq \emptyset$		
$e :\in (x \cup y)$		
$e \in x \wedge e \notin y$	$e \in x \wedge e \in y$	$e \notin x \wedge e \in y$
$z := z \mathbin{\widetilde{\cup}} f(\{e\}, \emptyset)$ $x := x \simeq e$	$z := z \mathbin{\widetilde{\cup}} f(\{e\}, \{e\})$ $x := x \simeq e$ $y := y \simeq e$	$z := z \mathbin{\widetilde{\cup}} f(\emptyset, \{e\})$ $y := y \simeq e$

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. □

Egyváltozós kétértékű eset

Legyen $f : X \to Y \times Z$ $(X, Y, Z \subseteq \mathfrak{F}(H), f_1 : X \to Y, f_2 : X \to Z, f = (f_1, f_2))$ elemenként feldolgozható függvény.

$$A = \underset{x}{X} \times \underset{y}{Y} \times \underset{z}{Z}$$
$$B = \underset{x'}{X}$$

$Q : (x = x')$

$R : (y = f_1(x') \wedge z = f_2(x'))$

Tétel: Ekkor az alábbi program megoldása a specifikált feladatnak:

$y, z := \emptyset, \emptyset$	
$x \neq \emptyset$	
	$e :\in x$
	$y, z := y \mathbin{\tilde{\cup}} f_1(\{e\}), z \mathbin{\tilde{\cup}} f_2(\{e\})$
	$x := x \simeq \{e\}$

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 terminálófüggvény marad, és a levezetés lépései is megegyeznek. □

Általános változat

Legyenek n, m rögzített természetes számok, $f : X_1 \times \cdots \times X_n \to Y_1 \times \cdots \times Y_m \ \ (X_i, Y_j \in \mathfrak{F}(H), \ \ (i \in [1..n], j \in [1..m]))$ 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 $f = (f_1, \ldots, f_m)$.

$$\begin{array}{lccccccccc} A = & X_1 & \times & \ldots & \times & X_n & \times & Y_1 & \times \ \ldots \ \times & Y_m \\ & x_1 & & & & x_n & & y_1 & & y_n \end{array}$$

$$\begin{array}{lccccc} B = & X_1 & \times & \ldots & \times & X_n \\ & x'_1 & & & & x'_n \end{array}$$

$Q : (x_1 = x'_1 \wedge \cdots \wedge x_n = x'_n)$

$R : (y_1 = f_1(x'_1, \ldots, x'_n) \wedge \cdots \wedge y_m = f_m(x'_1, \ldots, x'_n))$

Tétel: Ekkor az alábbi program megoldása a specifikált feladatnak:

$y_1, ..., y_m := \emptyset, ..., \emptyset$		
$\bigcup_{i=1}^{n} x_i \neq \emptyset$		
	$e :\in \bigcup_{i=1}^{n} x_i$	
$\setminus \ldots$	$\setminus \ \forall i \in I : e \in x_i \wedge \forall i \in [1..n] \setminus I : e \notin x_i$	$\setminus \ldots$
$\ldots$	$y_1, \ldots, y_m :=$ $y_1 \mathbin{\tilde{\cup}} f_1(s_1(e), \ldots, s_n(e)), \ldots, y_m \mathbin{\tilde{\cup}} f_m(s_1(e), \ldots, s_n(e))$ $\forall i \in I : x_i := x_i \simeq \{e\}$	$\ldots$

ahol $I \subseteq [1..n]$ és $I \neq \emptyset$,

$$\forall i \in [1..n] : s_i(e) = \begin{cases} \{e\}, & \text{ha } i \in I; \\ \emptyset, & \text{ha } i \notin I. \end{cases}$$

Az elágazás „ágainak” száma $2^n - 1$.
Bizonyítás: A tétel az alábbi invariáns tulajdonsággal és terminálófüggvénnyel könnyen levezethető.

$$P : (\forall j \in [1..m] : (y_j \cup f_j(x_1, \ldots, x_n) = f_j(x'_1, \ldots, x'_n) \wedge$$
$$y_j \cap f_j(x_1, \ldots, x_n) = \emptyset \wedge \forall i, k \in [1..n] : (x'_i \setminus x_i) \cap x_k = \emptyset))$$

$$t = \left| \bigcup_{i=1}^{n} x_i \right|$$

□

Megjegyzés: a fenti programozási tételtekből következik, hogy a 12.1 tétel feltétele nemcsak elégséges, de szükséges feltétele is az elemenkénti feldolgozhatóságnak.

12.5. Feladatok

12.1. Adott egy $f : \mathbb{Z} \to \mathbb{Z}$ függvény. Határozzuk meg, hogy a függvény melyik két pontban veszi fel a maximumát és a minimumát az $[m..n]$ intervallumban!

12.2. Határozzuk meg az x és az y természetes számok legnagyobb közös osztóját!

12.3. Határozzuk meg az x és az y természetes számok legkisebb közös többszörösét!

12.4. Határozzuk meg –a hatványozás műveletének használata nélkül – az x szám n-edik hatványát!

12.5. Döntsük el, hogy a k természetes szám osztja-e az x természetes számot!

12.6. Döntsük el, hogy az x természetes szám prímszám-e!

12.7. Adottak az x és y vektorok ($x.dom = y.dom$). Képezzük az $x + y$ és az $x - y$ vektorok skaláris szorzatát!

12.8. Határozzuk meg az x vektor elemeinek összegét úgy, hogy a páratlan indexű elemek a negáltjukkal szerepeljenek az összegzésben!

12.9. Adott egy egész számokból álló vektor és két egész szám. Állapítsuk meg, hogy a két szám előfordul-e a vektorban, és ha igen, akkor melyik előbb!

12.10. Adott az egész számokat tartalmazó x vektor. Permutáljuk a vektor elemeit (helyben!) úgy, hogy a vektor egy eleme a monoton rendezés szerinti helyére kerüljön, azaz ne előzze meg őt nála nagyobb elem, és utána ne legyen nála kisebb!

12.11. Adott az x négyzetes mátrix. Határozzuk meg az alsó háromszög elemeinek összegét!

12.12. Adott az x négyzetes mátrix. Tükrözzük (transzponáljuk) a mellékátlójára helyben (azaz az eredmény x-ben keletkezzen)!

12.13. Adott az x négyzetes mátrix. Tükrözzük (transzponáljuk) a főátlójára helyben (azaz az eredmény x-ben keletkezzen)!

12.14. Adott az x vektor. Számítsuk ki a b vektor ($b.dom \leq x.dom$) elemeinek értékét úgy, hogy b i-edik eleme az első i darab x-beli elem összege legyen!

12.15. Adottak az n és k számok. Számítsuk ki $\binom{n}{k}$ értékét!

12.16. Az x egész számokból álló vektor egy decimális szám számjegyeit tartalmazza helyi érték szerint csökkenő sorrendben. Számítsuk ki az ábrázolt szám értékét!

12.17. Adott egy természetes szám. Az x egészértékű vektorban állítsuk elő a szám számjegyeit helyi érték szerint csökkenő sorrendben, és adjuk meg azt is, hogy a szám hány számjegyből áll!

12.18. Az x egész számokból álló vektor egy decimális szám számjegyeit tartalmazza helyi érték szerint csökkenő sorrendben. Állítsuk elő x-ben az eredetinél eggyel nagyobb szám ugyanilyen ábrázolását, illetve mondjuk meg, ha túlcsordulás volt!

12.19. Az x egész számokból álló vektor egy decimális szám számjegyeit tartalmazza helyiérték szerint csökkenő sorrendben. Állítsuk elő x-ben az eredetinél eggyel kisebb szám ugyanilyen ábrázolását, illetve mondjuk meg, ha alulcsordulás volt!

12.20. Az azonos értelmezési tartományú x és y vektorok egy-egy $x.dom$ jegyű decimális szám számjegyeit tartalmazzák. A kisebb indexeken vannak 10 magasabb hatványainak együtthatói. Képezzük a z vektorban a számok összegét, illetve állapítsuk meg, hogy keletkezett-e túlcsordulás!

12.21. Adott az x vektor, melynek elemei k^2-es számrendszerbeli számjegyek. Állítsuk elő az így reprezentált szám k-s számrendszerbeli jegyeit az y vektorban (a szám magasabb helyi értékeit a vektor alacsonyabb indexű helyein találjuk)!

12.22. Adott az x vektor, melynek elemei k-s számrendszerbeli számjegyek. Állítsuk elő az így reprezentált szám k^2-es számrendszerbeli jegyeit az y vektorban (a szám magasabb helyi értékeit a vektor alacsonyabb indexű helyein találjuk)!

12.23. Egy vektor egy egész számot reprezentál úgy, hogy a vektor minden eleme a szám egy decimális számjegyét tartalmazza. Csökkentsük ezt a számot egy adott helyi értéken egy $0 - -9$ közötti értékkel!

12.24. Határozzuk meg az x természetes szám decimális alakja számjegyeinek számát!

12.25. Határozzuk meg az x természetes szám decimális alakja számjegyeinek összegét!

12.26. Adott egy egész számokból álló vektor. Rendezzük a vektor elemeit (helyben) csökkenő sorrendbe!

12.27. Adott az x és b vektor úgy, hogy b az x indexeiből veszi fel elemeit. Az x vektor minden b_j-edik elemének helyére írjunk nullát!

13. fejezet

Transzformációk

Ebben a fejezetben azzal fogunk foglalkozni, hogyan lehet a programozási tételeket konkrét feladatok esetében alkalmazni.

Először egy olyan feladatot nézünk meg, amelynek látszólag semmi köze a programozáshoz: oldjuk meg az $x^2 - 5x + 6 = 0$ egyenletet. Az egyik lehetőség az, hogy valamilyen gondolatmenettel (gyökök és együtthatók közötti összefüggés vagy teljes négyzetté alakítás) $(x-3)(x-2) = 0$ alakra hozzuk, és azután megoldjuk az $x-3 = 0$ és az $x - 2 = 0$ egyenleteket. Vagyis az eredeti feladat helyett két másik, de egyszerűbb feladatot kell megoldani. Pontosan ez a *levezetés* lényege is. Ezt a módszert alkalmaztuk az előző fejezetben.

A másik lehetőség az, hogy megoldjuk a következő általános feladatot: $ax^2 + bx + c = 0$. Belátjuk azt a tételt, hogy ha $a \neq 0$, akkor a gyököket a közismert gyökképlettel kapjuk meg:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Ezután alkalmazzuk a tételt: a képletben a helyébe 1-et, b helyébe -5-öt és c helyébe 6-ot helyettesítve megkapjuk a gyököket. A programozásban ezt a módszert nevezzük *visszavezetésnek*.

A másodfokú egyenlet gyökképletét gyakran alkalmazhatjuk nem másodfokú egyenletek megoldása esetén is, például legyen a megoldandó egyenlet: $x^4 - 5x^2 + 4 = 0$. Az $x^2 = y$ helyettesítést alkalmazva y-ra kapunk egy másodfokú egyenletet, amit megoldva már csak az $x^2 = 1$ és $x^2 = 4$ egyenleteket kell megoldanunk. Azt mondhatjuk, hogy egy *transzformáció* segítségével oldottuk meg a feladatot.

A továbbiakban először a visszavezetéssel foglalkozunk.

13.1. Visszavezetés

A legegyszerűbb esetben a megoldandó feladat és a programozási tételek állapottere azonos, legfeljebb a jelölésben különbözik, és átjelölés után az elő- és utófeltételek is megegyeznek. Ekkor a megoldóprogramot is egyszerű átjelöléssel kapjuk meg. Ez az eset azonban elég ritka, erre szorítkozva a visszavezetés használhatósága nagyon korlátozott lenne.

Emlékeztetünk azonban arra, hogy a megoldás 5.1. definíciója szerint a feladat és a program állapottere különböző is lehet; másrészt felhasználhatjuk azt az összefüggést,

hogy egy szigorúbb feladat megoldása egyben megoldása a gyengébb feladatnak is, lásd a 2.1. állítást.

Ebből következik, hogy ha a tétel utófeltétele szigorúbb, mint a feladaté, vagy ha a tétel állapotterében szerepelnek olyan komponesek, amelyek a feladatéban nem (a kiterjesztés 4.1. definíciója szerint a tétel ebben az esetben is szigorúbb), akkor a tétel programja, esetleges átjelölés után, megoldása lesz a feladatnak. Ilyen esetekben mondjuk azt, hogy *szigorítással* vezetünk vissza.

Ha a feladat állapottere tartalmaz olyan komponenseket, amelyek nincsenek benne a tétel állapotterében, így a 4.2. definíció szerint a tétel megoldóprogramja nem változtatja meg őket, az ilyen változóknak a feladat elő- és utófeltételében egy-egy paraméterváltozóval rögzítettnek kell lenniük. Az ilyen változókat a *visszavezetés paramétereinek* nevezzük, és rendszerint a tétel „adott függvényeit” határozzák meg.

Előfordul, hogy a tétel feladatának értelmezési tartománya szűkebb, mint a megoldandó feladaté, ebben az esetben egy *feltétel* segítségével leszűkítjük az értelmezési tartományt. Valójában itt levezetési lépést alkalmazunk, a feladat megoldása egy elágazás, ha a feltétel teljesül, a megoldóprogram a tétel programja, egyébként vagy más tételt, vagy levezetést alkalmazunk.

13.2. Egyszerű programtranszformációk

A következő transzformációk segítségével olyan, az eredetivel ekvivalens programot hozunk létre, amely valamilyen szempontból előnyösebb számunkra, mint az eredeti. A szempont alapvetően kétféle lehet. Az egyik: a transzformált program megengedett programkonstrukció, míg az eredeti nem az. Gyakran fordul elő, hogy visszavezetéssel nem megengedett programot kapunk. A másik szempont is a visszavezetéshez kapcsolódik: sokszor a levezetéssel kapott program nagyon ügyetlen, és a transzformációval ezen javítunk.

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:

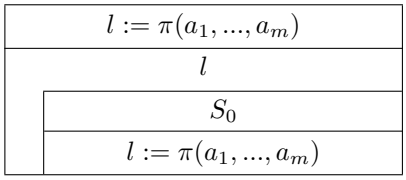
$$A' = \underset{a_1}{A_1} \times \ldots \times \underset{a_m}{A_m} \times \underset{l_1}{\mathbb{L}} \times \ldots \times \underset{l_n}{\mathbb{L}}$$

$l_1, \ldots, l_n := \pi_1(a_1, ..., a_m), \ldots, \pi_n(a_1, ..., a_m)$
$IF(l_1 : S_1, ..., l_n : S_n)$

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

Nem megengedett ciklusfeltétel kitranszformálása. Legyen $S \subseteq A \times A^{**}, S = DO(\pi : S_0), A = A_1 \times ... \times A_m$. Konstruáljuk az $S' \subseteq A' \times A'^{**}$ programot az alábbi módon:

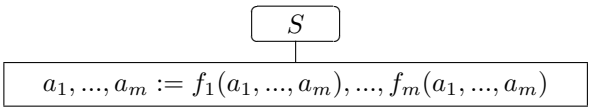
$$A' = \underset{a_1}{A_1} \times \ldots \times \underset{a_m}{A_m} \times \underset{l}{\mathbb{L}}$$



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

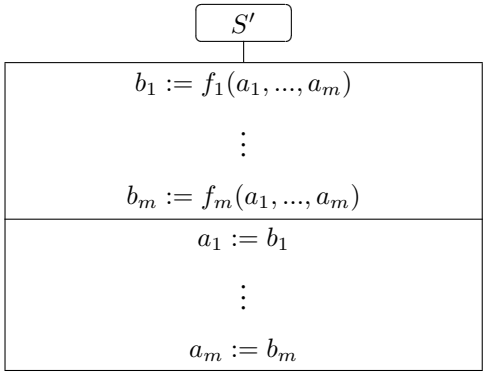
Szimultán értékadás helyettesítése egyszerű értékadásokkal. Legyen $S \subseteq A \times A^{**}$ a következő szimultán értékadás:

$$A = \underset{a_1}{A_1} \times \ldots \times \underset{a_m}{A_m}$$



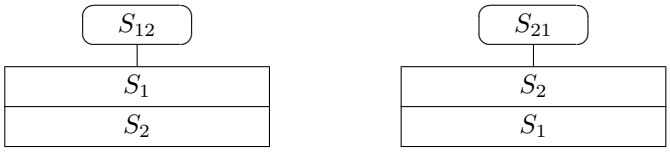
Konstruáljuk az $S' \subseteq A' \times A'^{**}$ programot az alábbi módon:

$$A' = \underset{a_1}{A_1} \times \ldots \times \underset{a_m}{A_m} \times \underset{b_1}{A_1} \times \ldots \times \underset{b_m}{A_m}$$

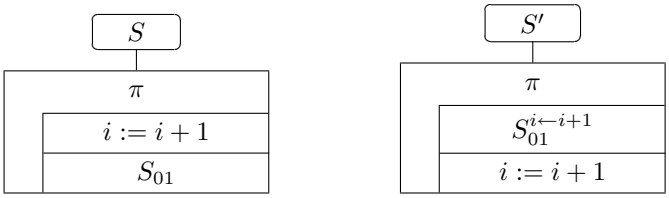


Ekkor az S' program ekvivalens S-sel A-n.

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}-gyel.



Ciklusmagbeli 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 definiált 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:

$$\begin{array}{ccccccc} A' & = & A_1 & \times & \ldots & \times & A_m & \times & H \\ & & a_1 & & & & a_m & & z \end{array}$$

$z := f(a_{i_1}, ..., a_{i_k})$
$S^{f(a_{i_1},...,a_{i_k})\leftarrow z}$

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, $t_0, t_{-1}, \ldots, t_{-k+1} \in H$ rögzített, és definiáljuk az $f : \mathbb{Z} \to H$ parciális függvényt az alábbi módon:

$$\begin{array}{rcl} f(0) & = & t_0, \\ f(-1) & = & t_{-1}, \\ \vdots & & \vdots \\ f(-k+1) & = & t_{-k+1}, \end{array}$$

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:

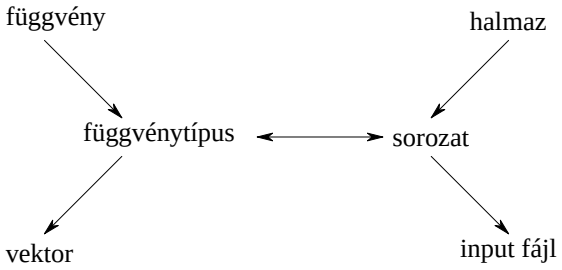
$i, z, z_{-1}, .., z_{-k+1} := 0, t_0, t_{-1}, .., t_{-k+1}$
S_1
π
$z, z_{-1}, .., z_{-k+1} := F(i+1, f(i), ..., f(i-k+1)), z, .., z_{-k+2}$
${S_0}^{f(i+1)\leftarrow z}$
$i := i + 1$

Mindegyik esetben egyszerűen belátható a programfüggvények azonossága.

13.3. Típustranszformációk

Típustranszformációról akkor beszélhetünk, ha az állapottér bizonyos komponenseit valami kapcsolódó típusra cseréljük.

A programozási tételek – és általában véve a (feladat, megoldóprogram) 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.



13.1. ábra. Típusok közötti kapcsolatok

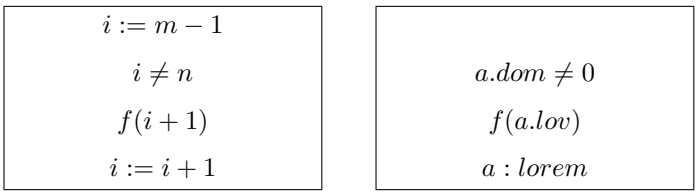
A programozási tételek átírása más típusra. Az alábbi sémák a programozási tételek típusok közötti transzformációjakor elvégzendő helyettesítéseket definiálják. A transzformáció úgy történik, hogy az első típus adott műveletét a második típus megfelelő (azonos sorban levő) műveletére cseréljük.

Induljunk el az intervallumon értelmezett függvényektől. A függvénytípusra való áttérés lényegében nem igényel transzformációt, hiszen az $f : func(\mathbb{Z}, H)$ értéke egy $[lob(f)..hib(f)]$ intervallumon értelmezett $\mathbb{Z} \rightarrow H$ függvény. Ha olyan programról van szó, ahol a függvény rögzített – a tételeink ilyenek –, akkor egyúttal vektorra is transzformáltunk.

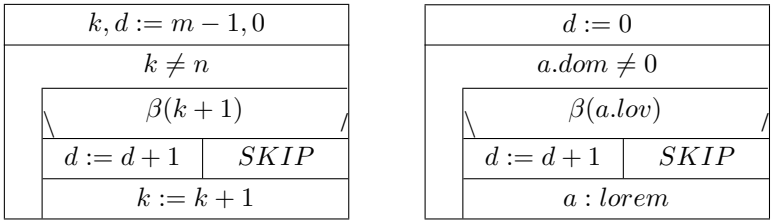
- **Intervallum felett definiált függvényről sorozatra**

  Az $s : seq(H)$ sorozat is tekinthető az $[1..dom(s)]$ intervallumon értelmezett $\mathbb{Z} \rightarrow H$ függvénynek, de a műveletek között nem szerepel az indexelés. Ezért nem is minden programunk írható át sorozatra. Tegyük föl, hogy egy programban mindig csak az $f(i+1)$-re hivatkozunk, ahol f az $[m..n]$ intervallumon értelmezett függvény. Az első hivatkozást megelőzően $i := m - 1$, utána pedig az $i := i + 1$ programok kivételével minden más programban i konstans. Ha

ezek a feltételek teljesülnek, akkor garantálható a következő invariáns tulajdonság: $s =< f(i+1), \ldots, f(n) >$. Azaz a következő helyettesítéssel transzformálhatjuk a programot:

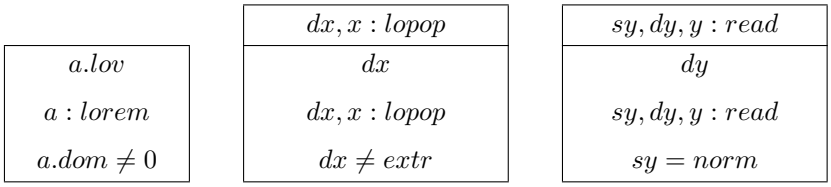


Példa (számlálás tétele):



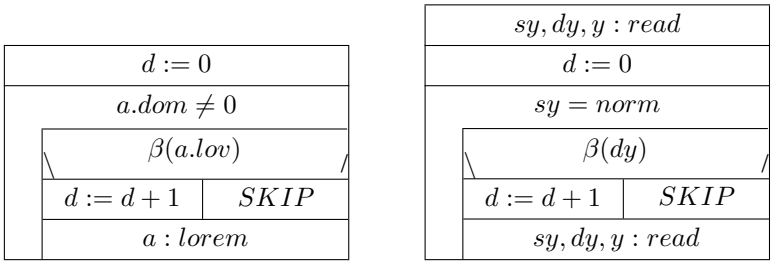
- **Sorozatról szekvenciális input fájlra** ($dx, x : lopop$ vagy $sy, dy, y : read$)

  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.



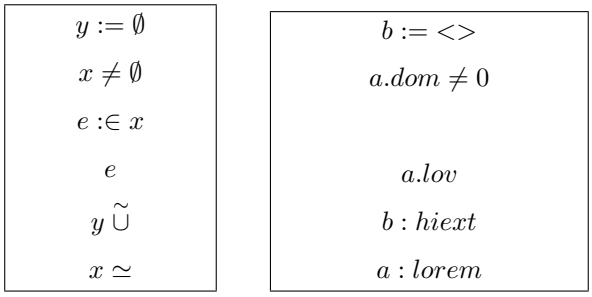
  A transzformált program egy plusz $lopop$ (vagy $read$) művelettel kezdődik, ezért ezt a transzformációt *előreolvasási technikának* nevezzük.

  Példa (számlálás tétele):



- **Halmazról sorozatra** (egy input halmaz/sorozat esetén)

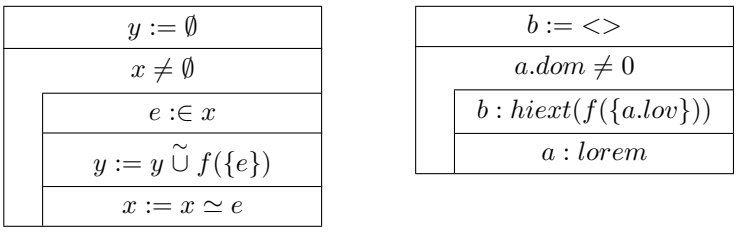
  A két típus közötti megfeleltetés: az a és b sorozatok tagjai felsorolják az x és y halmazok elemeit.



A fenti megfeleltetést alkalmazhatjuk az elemenkénti feldolgozás esetén is, feltéve az f függvényről, hogy egy egyelemű halmazhoz egyelemű halmazt rendel hozzá. Ez viszonylag ritkán teljesül, az azonban már nagyon gyakori, hogy legfelebb egyelemű a kép. Ebben az esetben $b : hiext$ helyébe egy elágazás, $IF(d = \emptyset : SKIP, d \neq \emptyset : (b : hiext(d)))$ kerül.

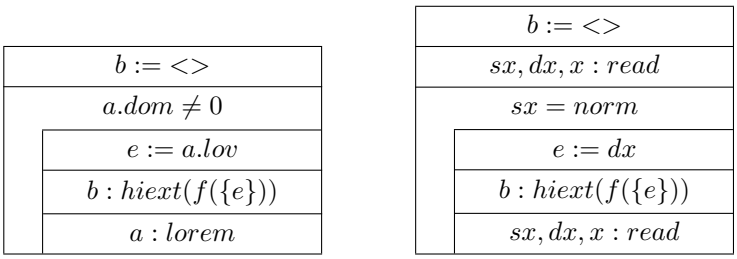
Megjegyezzük, hogy az $e :\in x$ művelet helyett az $e := a.lov$ értékadást is írhattuk volna, ami szintén szigorítást jelent az eredeti értékkiválasztáshoz képest.

Példa (egyváltozós egyértékű elemenkénti feldolgozás):



Az előzőekben tárgyalt előreolvasási technika segítségével megkaphatjuk a szekvenciális fájlra vonatkozó változatot is.

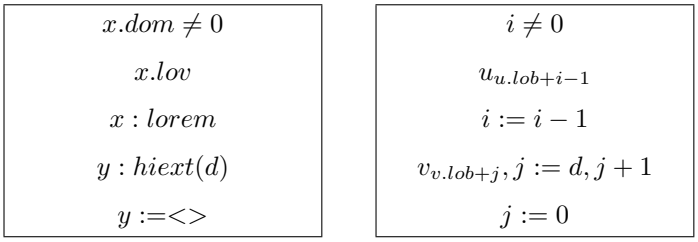
Példa (egyváltozós egyértékű elemenkénti feldolgozás):



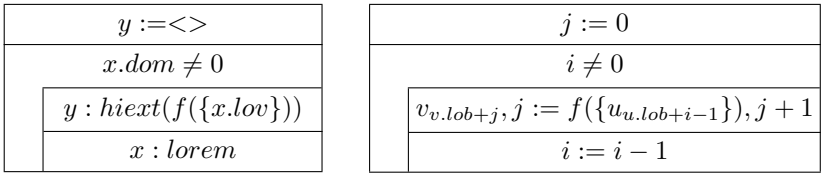
- **Sorozatról vektorra**

  A transzformációt az akadályozza, hogy a vektor értelmezési tartományát nem változtathatjuk meg. Ezt a problémát könnyen kezelhetjük input sorozat esetén, de az output sorozat esetében a transzformáció csak akkor hajtható végre, ha föltesszük, hogy a sorozat elemeinek száma nem haladja meg a vektor értelmezési tartományának számosságát.

  Feleltessük meg az x (és y) sorozatnak az (u, i) (illetve (v, j)) párt, ahol u, v vektorok és i, j nemnegatív egészek, oly módon, hogy i az x (illetve j az y) sorozat elemeinek száma, és az elemei az u vektor $u_{u.lob}, \ldots, u_{u.lob+i-1}$ (illetve a v vektor $v_{v.lob}, \ldots, v_{v.lob+i-1}$) elemeivel egyeznek meg.

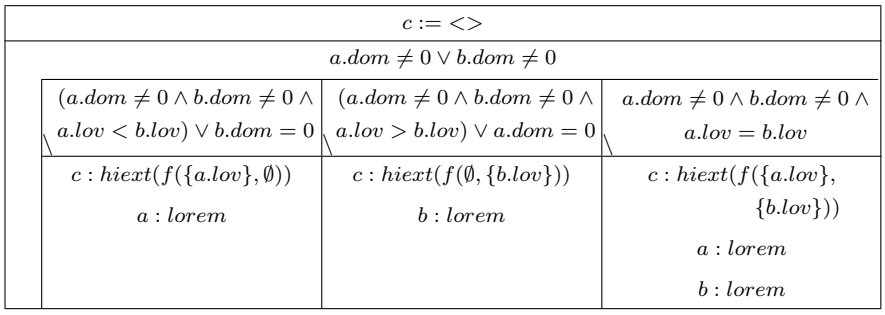


Példa (egyváltozós egyértékű elemenkénti feldolgozás):



- **Halmazokról sorozatokra**

Kettő (több) input halmaz a tárgyalt tételek között a két(több)változós elemenkénti feldolgozás esetében fordul elő. A kétváltozós elemenkénti feldolgozásnál az okozhat gondot, hogy el kell tudnunk dönteni, egy adott elem melyik sorozatban van benne. Ezért feltesszük, 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.



A teljesség kedvéért bemutatjuk a szekvenciális fájlokra vonatkozó változatot is, alkalmazva az előreolvasási technikát:

<table>
<tr><td colspan="3">$sx, dx, x : read$
$sy, dy, y : read$</td></tr>
<tr><td colspan="3">$c := <>$</td></tr>
<tr><td colspan="3">$sx = norm \vee sy = norm$</td></tr>
<tr><td>$sy = abnorm \vee$
$sx = sy \wedge dx < dy$</td><td>$sx = abnorm \vee$
$sx = sy \wedge dx > dy$</td><td>$sx = sy \wedge$
$dx = dy$</td></tr>
<tr><td>$c : hiext(f(\{dx\}, \emptyset))$
$sx, dx, x : read$</td><td>$c : hiext(f(\emptyset, \{dy\}))$
$sy, dy, y : read$</td><td>$c : hiext(f(\{dx\}, \{dy\}))$
$sx, dx, x : read$
$sy, dy, y : read$</td></tr>
</table>

Megjegyezzük, hogy a feltételekben azért írhattunk $sx = norm \wedge sy = norm$ helyett $sx = sy$-t, mert a ciklusfeltétel szerint nem lehet mindkettő $abnorm$.

13.4. 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.

13.4.1. 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 állapottér-transzformációt 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. Legyen

$$E = \bigcup_{i=1}^{n} E_i \text{ és } F = \{E_1, E_2, \ldots, E_n\}.$$

Tegyük fel, hogy a $\mathcal{T}$ 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. 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 név részeinek sorozata ($\in NEV^*$);
$seq(t\|CHAR)$	a t sorozatbeli rekordok név részének 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 név részeiből képzett karaktersorozat;
$seq(t\|R)$	üres sorozat.

13.5. Példa állapottér-transzformációra

Tegyük fel, hogy van egy karakterekből álló szekvenciális fájlunk, ami egy szöveget tartalmaz. A feladat az, hogy számoljuk meg, hány olyan csupa betűből álló rész van a szövegben, amelynek hossza nagyobb, mint egy megadott k érték!

Így első ránézésre a feladat nem vezethető vissza egyetlen ismert programozási tételre sem. A feladatot ezért úgy transzformáljuk, hogy bevezetünk egy új állapotteret. Tegyük fel, hogy a szövegfájl helyett egy olyan fájlunk van, amelyben szavak – betűkből álló sorozat – és elválasztó részek – csupa nem betű karakterből álló sorozat – vannak. Azaz az eredeti állapottér:

$$\begin{array}{lll} A = & \underset{x}{X} & \times \quad \underset{d}{\mathbb{N}} \ , \end{array} \qquad X = file(CH),$$
$$Y = seq(\mathbb{N}),$$

és az új állapottér:

$$\begin{array}{lll} A' = & \underset{f}{F} & \times \quad \underset{d}{\mathbb{N}} \ , \end{array} \qquad F = file(\textit{SZÓ;ELVR}),$$
$$\textit{SZÓ} = seq^{+}(\textit{BETŰ}), \qquad \textit{ELVR} = seq^{+}(\textit{NBETŰ}),$$

ahol *BETŰ* jelenti a betű, és *NBETŰ* a nem betű karakterek halmazát. Az F-re még egy invariáns tulajdonságot is fölírunk. Két egymást követő elem típusa nem lehet azonos, vagyis

$$I_F(z) = (\forall i \in [1..dom(z) - 1] : x_i.\textit{SZÓ} \neq zi + 1.\textit{SZÓ}).$$

Mi a kapcsolat a két állapottér között? Természetesen az, hogy x és f ugyanazokból a karakterekből áll, és a sorrendjük is azonos. Ezt fejezi ki az, hogy

$$seq(x|\{\textit{BETÜ,NBETŰ}\}) = seq(f|\{\textit{BETŰ,NBETŰ}\}).$$

Most már meg tudnánk fogalmazni, hogy a szavak hosszait akarjuk meghatározni, de ehhez az elválasztó részekre nincs is szükség, ezért azokat el is hagyjuk.

$$\begin{array}{lll} A'' = & \underset{g}{G} & \times \quad \underset{d}{\mathbb{N}} \ , \end{array} \qquad F = file(\textit{SZÓ}),$$

és az f és g közötti kapcsolat

$$seq(f|\{\textit{SZÓ}\}) = seq(f|\{\textit{SZÓ}\}).$$

Mivel nekünk nem a szavakra, hanem csak a hosszukra van szükségünk, tovább transzformáljuk az állapotteret.

$$\begin{array}{lll} A''' = & \underset{h}{H} & \times \quad \underset{d}{\mathbb{N}} \end{array} \qquad F = file(\mathbb{N})$$

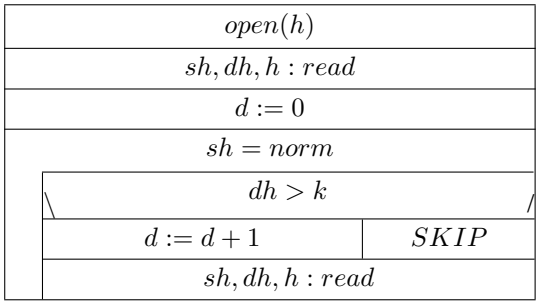
és a g és h közötti kapcsolat

$$dom(h) = dom(g) \text{ és } \forall i \in [1..dom(g) : h_i = dom(g_i).$$

Tehát a feladat specifikációja:

$$\begin{array}{lll} A = & \underset{h}{H} & \times \quad \underset{d}{\mathbb{N}} \end{array}$$
$$\begin{array}{lll} B = & \underset{h'}{H} \end{array}$$
$$Q : (h = h')$$
$$R : (d = \sum_{i=1}^{dom(h)} \chi(h_i > k))$$

Ez a feladat visszavezethető a számlálás tételének szekvenciális fájlra felírt változatára:

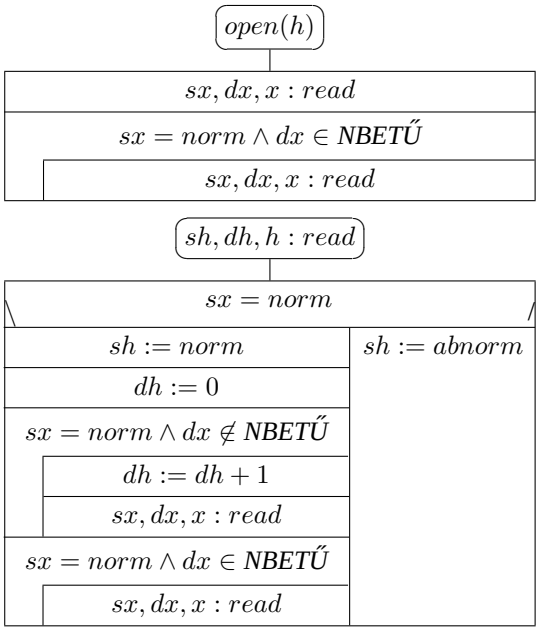


Hátra van még az absztrakt $open$ és $read$ műveletek megvalósítása.

Az absztrakt h fájlnak pontosan akkor van még eleme, ha az eredeti x fájlban van még karakter. Ezért van szükség az $open$ műveletre, amely arra hivatott, hogy biztosítsa a $read$ művelet elején megkívánt invariáns tulajdonságot: vagy $sx = abnorm$ vagy dx 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 fájl is kiürült.

Az absztrakt fájlok 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. *Az absztrakt* $read$ mindig egy elágazás a $read$ definíciójának megfelelően.

Nézzük tehát az absztrakt műveletek megvalósítását az eredeti állapottéren: ezek, mint látható, szintén programozási tételek egyszerű alkalmazásai:



13.6. Programinverzió

13.6.1. Egyváltozós eset

Legyenek C, D, E és F tetszőleges típusok. Legyen továbbá $X = seq(E)$ és $Y = seq(F)$, valamint $f_1 : C \to X$, $f_2 : X \to Y$ és $f_3 : Y \to D$ függvények.

Tekintsük az alábbi specifikációval adott feladatot:

$$\begin{array}{lcl} A = & C & \times \ D \\ & c & \quad d \end{array}$$
$$\begin{array}{ll} B = & C \\ & c' \end{array}$$
$$Q : (c = c')$$
$$R : (d = f_3 \circ f_2 \circ f_1(c'))$$

Tegyük fel, hogy az f_1 függvény értékét kiszámító program az alábbi alakban írható fel:

$$\begin{array}{lcl} A = & C & \times \ X \\ & c & \quad x \end{array}$$
$$\begin{array}{ll} B = & C \\ & c' \end{array}$$
$$Q : (c = c')$$
$$R : (x = f_1(c'))$$

$S_{11}(c)$	
$x :=<>$	
π	
	$S_{12}(c, e)$
	$x : hiext(e)$
$S_{13}(c)$	

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:

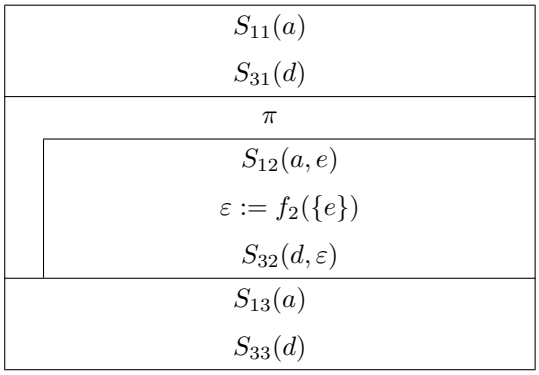
$$\begin{array}{lcl} A = & Y & \times \ D \\ & y & \quad d \end{array}$$
$$\begin{array}{ll} B = & Y \\ & y' \end{array}$$
$$Q : (y = y')$$
$$R : (d = f_3(y'))$$

$S_{31}(d)$	
$y.dom \neq 0$	
	$S_{32}(d, y.lov)$
	$y : lorem$
$S_{33}(d)$	

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 következő programtranszformációval, melyet *programinverziónak* hívunk. A közbülső két sorozat típust kihagyjuk, és a három ciklust egybeírjuk az alábbi módon:



13.6.2. 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:

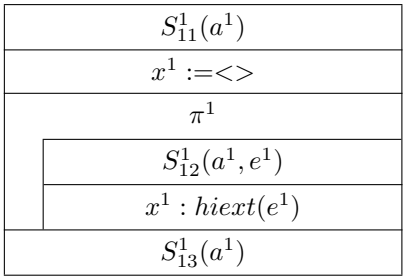
$$\begin{array}{lccccc} A = & A^1 & \times & A^2 & \times & D \\ & a^1 & & a^2 & & d \end{array}$$

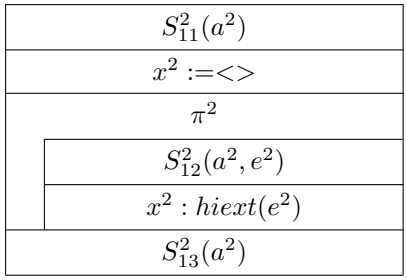
$$\begin{array}{lccc} B = & A^1 & \times & A^2 \\ & a^{1\prime} & & a^{2\prime} \end{array}$$

$$Q : (a^1 = a^{1\prime} \wedge a^2 = a^{2\prime})$$

$$R : (d = f_3 \circ f_2(f_1^1(a^{1\prime}), f_1^2(a^{2\prime})))$$

Legyenek az f_1^1 és f_1^2 függvényeket kiszámító elemenként előállító programok az alábbiak:





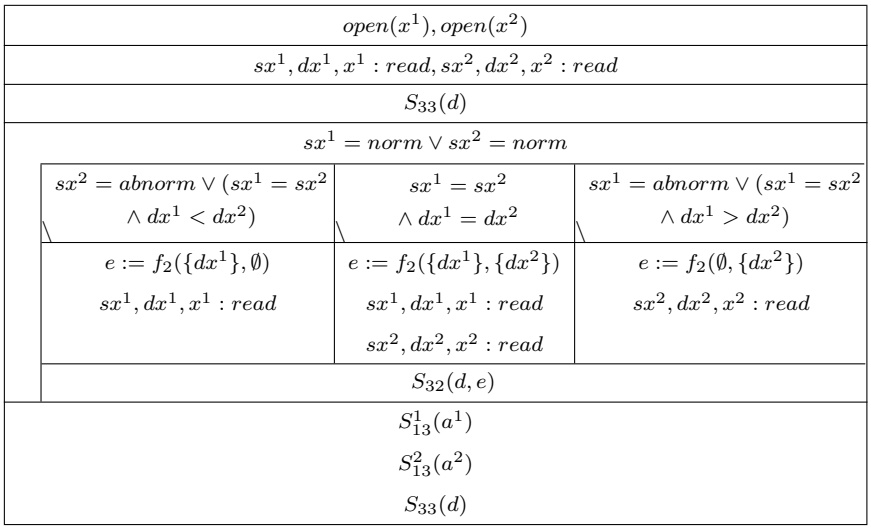
és tegyük fel, hogy az előállított x^1 és x^2 sorozatok rendezettek.

Legyen f_3, mint korábban, elemenként felhasználó prgrammal kiszámított függvény. Tegyük fel, hogy f_2 egy olyan kétváltozós, egyértékű elemenként feldolgozható

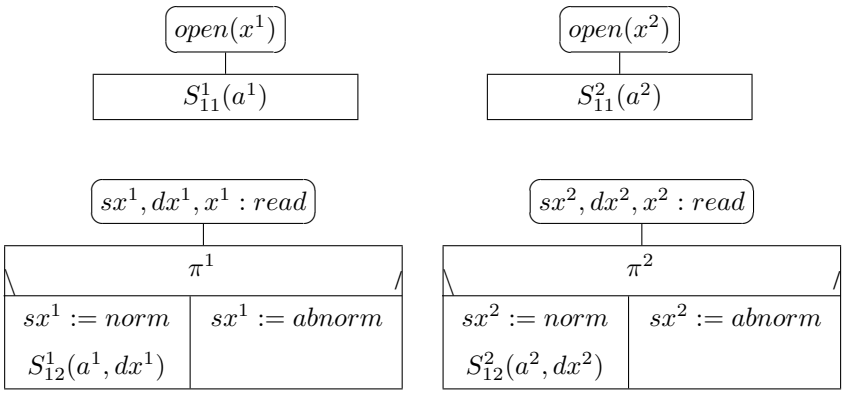
függvény, amely minden olyan halmazpárhoz, amelynek tagjai legfeljebb egy elemet tartalmaznak, egy egyelemű halmazt rendel hozzá.

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 fájloknál felmerülttel analóg: az elemet előállító program (S_1^2, illetve S_1^1) az egyes elemeket ugyanúgy szolgáltatja, mint ha fájlból olvasnánk őket: csak akkor nézhetjük meg a következő elemet, ha legeneráltatjuk. Ez sugallja azt a megoldást, hogy az x^1 és x^2 sorozatokat tekintsük absztrakt fájloknak az elemenként feldolgozásban. Az elemenként felhasználó program beinvertálása nem okoz gondot, ugyanúgy történik, mint az egyváltozós esetben.



ahol az absztrakt fájl műveleteinek megvalósításai:



13.7. Példák

13.1. példa: Keressük meg az $f : \mathbb{Z} \to \mathbb{Z}$ függvény olyan értékeinek a maximumát az $[a, b]$ intervallumon (és egy hely indexét), ahol az argumentum és a hozzá tartozó érték paritása azonos!
Megoldás:

$$\begin{array}{lccccccc} A = & \mathbb{Z} & \times & \mathbb{Z} & \times & \mathbb{Z} & \times & \mathbb{L} \\ & a & & b & & i & & l \end{array}$$

$$\begin{array}{lccc} B = & \mathbb{Z} & \times & \mathbb{Z} \\ & a' & & b' \end{array}$$

$Q : (a = a' \wedge b = b' \wedge a \leq b + 1)$
$R : (Q \wedge l = (\exists i \in [a..b] : 2|i + f(i)) \wedge l \to (i \in [a..b] \wedge 2|i + f(i) \wedge max = f(i) \wedge \forall j \in [a..b] : (2|j + f(j)) \to (f(j) \leq f(i))))$

A specifikáció nagyon hasonló a feltételes maximumkeresés programozási tételéhez. Az eltéréseket az alábbi táblázattal foglalhatjuk össze:

feladat		feltételes maximumkeresés
a	$\leftrightarrow$	m
b	$\leftrightarrow$	n
$2\|i + f(i)$	$\leftrightarrow$	$\beta(i)$

Az olyan feladatokat, amelyek specifikációja csak átnevezéseket tesz egy már ismert programozási tétel specifikációjához képest, *természetes visszavezetés*sel oldhatjuk meg. Az említett átnevezés érintheti a tétel változóinak neveit, az általános kifejezéseket (pl. $\beta(i)$-t) vagy az utófeltétel állandó értékű kifejezéseit. Ismert programozási tételeknek tekintjük a programozási tételek különböző átírásait (pl. vektorra, sorozatra, input fájlra).

Amennyiben a specifikációkat ily módon sikerül egy alakra hoznunk, akkor az átnevezéseket a programozási tétel már ismert programján elvégezve megkapjuk a kitűzött feladatot megoldó programot.

<table>
<tr><td colspan="4">$k, l := a - 1, hamis$</td></tr>
<tr><td colspan="4">$k \neq b$</td></tr>
<tr><td>$2 \not| (k + 1 + f(k + 1))$</td><td>$2|(k + 1 + f(k + 1)) \wedge \neg l$</td><td colspan="2">$2|(k + 1 + f(k + 1)) \wedge l$</td></tr>
<tr><td rowspan="2">$SKIP$</td><td rowspan="2">$l, i, max := igaz, k + 1, f(k + 1)$</td><td>$f(k + 1) \geq max$</td><td>$f(k + 1) \leq max$</td></tr>
<tr><td>$i, max := k+1, f(k+1)$</td><td>$SKIP$</td></tr>
<tr><td colspan="4">$k := k + 1$</td></tr>
</table>

13.2. példa: Adjunk meg egy, az n és $2n$ természetes számok közé eső prímszámot!
Megoldás:

$$\begin{array}{lccc} A = & \mathbb{N} & \times & \mathbb{N} \\ & n & & p \end{array}$$

$$\begin{array}{lc} B = & \mathbb{N} \\ & n' \end{array}$$

$Q : (n = n' \wedge \exists j \in [n..2n] : \text{prím}(j))$
$R : (Q \wedge p \in [n..2n] \wedge \text{prím}(i))$

A lineáris keresés 1. változatának specifikációja hasonló. Lássuk, hogy az alábbi átnevezések után milyen különbségek maradnak!

feladat		lineáris keresés 1.0
n	$\leftrightarrow$	m
p	$\leftrightarrow$	i
prím(i)	$\leftrightarrow$	$\beta(i)$

Különbség, hogy az előfeltételünk szigorúbb, hiszen a j futóindexnek nem csupán n fölött, hanem $2n$ alatt kell lennie. (Megjegyezzük, hogy a specifikáció előfeltétele az általánosságot nem szorítja meg, ugyanis minden n természetes szám esetén létezik egy prím n és $2n$ között (Csebisev-tétel).

A lineáris keresés 1. utófeltételének behelyettesítés utáni alakja:
$R' = (Q \wedge p \geq n \wedge \text{prím}(p) \wedge \forall j \in [n..p-1] : \neg\text{prím}(j))$

Ez az utófeltétel biztosítja az R feltételt a Csebisev-tétel miatt. Hiszen, ha p-nek olyannak kell lennie, hogy $p-1$-ig ne legyen prím n-től, és p prím, akkor p biztosan $2n$-nél kisebb. Ugyanakkor az R nem követeli meg, hogy $p-1$-ig minden szám n-től kezdve összetett legyen, ezért a két feltétel között egyenlőség nem, csupán következés áll fenn ($R' \Rightarrow R$).

Tehát a tétel szigorúbb, mint a feladatunk, ezért szigorítással vezetünk vissza.

$p := n$	
$\neg$prím(p)	
	$p := p + 1$

13.3. példa: Határozzuk meg az n természetes szám osztóinak a számát!
Megoldás:

$$A = \begin{matrix} \mathbb{N} \\ n \end{matrix} \times \begin{matrix} \mathbb{N}_0 \\ d \end{matrix}$$

$$B = \begin{matrix} \mathbb{N} \\ n' \end{matrix}$$

$Q : (n = n')$

$R : (Q \wedge d = \sum_{i=1}^{n} \chi(i|n))$

A specifikáció nagyon hasonló a számlálás programozási tételéhez. Az eltéréseket az alábbi táblázattal foglalhatjuk össze:

feladat		számlálás
1	$\leftrightarrow$	m
$i\|n$	$\leftrightarrow$	$\beta(i)$

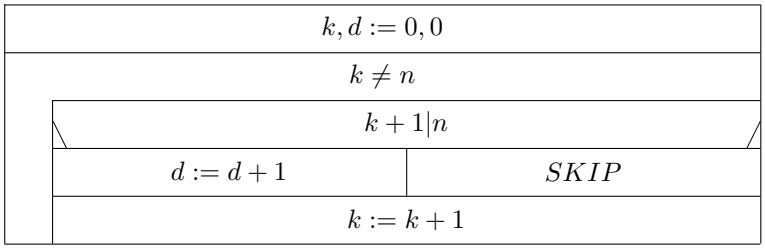
Ez a visszavezetés majdnem természetes visszavezetés. Bökkenő, hogy a felhasználni kívánt programozási tétel specifikációjában szereplő állapottér bővebb a feladatunk állapotterénél. Tehát a mi feladatunk állapottere altere a programozási tétel állapotterének. Ha a feladatunkat kiterjesztjük a tétel állapotterére, és az előfeltételt kibővítjük az $m = 1$ feltétellel, akkor ezzel a tétel feladatának értelmezési tartományát szűkítettük le, azaz megint szigorítással vezethetünk vissza.

A fenti tulajdonsággal rendelkező visszavezetést *alteres* visszavezetésnek, azon belül *konstanssal helyettesítésnek* is nevezzük. Erről akkor beszélünk, ha a feladat ál-

lapotteréből olyan komponens hiányzik, amit a programozási tétel programja nem változtatna meg. Ekkor ugyanis az a változó nyilván helyettesíthető akár egy konstanssal is a visszavezetés során.

Ellenőrizni szükséges, hogy a konstans érték, amellyel m-et helyettesítettük, teljesíti-e a számlálás előfeltételének rá vonatkozó nem triviális részét ($(m \leq n+1) = (1 \leq n+1)(n \in \mathbb{N})\surd$).

Ezekkel a megjegyzésekkel most már felírhatjuk a megoldóprogramot:



13.4. példa: Állapítsuk meg, hogy van-e az $f : \mathbb{Z} \to \mathbb{Z}$ függvény értékei között páros szám az $[m..n]$ intervallumban!

Megoldás:

$$\begin{array}{lccccc} A = & \mathbb{Z} & \times & \mathbb{Z} & \times & \mathbb{L} \\ & m & & n & & l \end{array}$$

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

$Q : (n = n' \wedge m = m' \wedge m \leq n+1)$

$R : (Q \wedge l = (\exists j \in [m..n] : 2|f(j)))$

A lineáris keresés 2.8 specifikációja hasonló. Lássuk, hogy az alábbi átnevezés után milyen különbségek maradnak:

feladat		lineáris keresés 2.8
$2\|f(i)$	$\leftrightarrow$	$\beta(i)$

A különbség az, hogy az i-t, azaz azt az eredménykomponenst, ami mutatná a megtalált páros elem helyét, az állapotterünk nem is tartalmazza. Ez a különbség persze indukálja az utófeltétel összes olyan tagjának kiesését is, amely i-re tett kikötést.

Ebben az esetben a feladatunk kiterjesztésénél a tétel nyílván szigorúbb, így szigorítással vezetünk vissza.

Ezt a visszavezetést is nevezhetjük *alteres visszavezetésnek*, hiszen a feladatunk állapottere altere a programozási tétel állapotterének. Ugyanakkor itt most nem arról van szó, hogy azt a komponenst konstanssal helyettesítettük volna. Amikor a programozási tétel állapotterének egy eredménykomponensét (tehát aminek kezdőértéke tetszőleges, de az utófeltétel tesz rá kikötést) nem találjuk meg a feladat állapotterében (és így természetesen az utófeltételében sem), akkor az alteres visszavezetés *általánosított* esetéről beszélünk.

Természetesen a feladatot ekkor is úgy oldjuk meg, hogy a programozási tételben írjuk át a táblázatunkban összeszedett változtatásokat, és végeredményben a megoldóprogram nem fog kisebb állapottéren futni. Azt, hogy a program más eredményeket is elér, amelyre mi nem vagyunk kíváncsiak, megtűrjük.

$i, l := m - 1, hamis$
$\neg l \wedge i \neq n$
$l := 2 \mid f(i+1))$
$i := i + 1$

13.5. példa: Keressük meg az $[m..n]$ intervallumban azt a legkisebb k számot, amelyre p és k relatív prímek!
Megoldás:
$A = \begin{array}{ccccccccc} \mathbb{N} & \times & \mathbb{N} & \times & \mathbb{N} & \times & \mathbb{L} & \times & \mathbb{N} \\ m & & n & & k & & l & & p \end{array}$
$B = \begin{array}{ccccc} \mathbb{N} & \times & \mathbb{N} & \times & \mathbb{N} \\ m' & & n' & & p' \end{array}$
$Q : (n = n' \wedge m = m' \wedge m \leq n + 1 \wedge p = p')$
$R : (Q \wedge l = (\exists j \in [m..n] : lnko(p, j) = 1) \wedge l \rightarrow (k \in [a..b] \wedge lnko(p, k) = 1 \wedge \forall j \in [m..k-1] : lnko(p, j) \neq 1))$

A specifikáció nagyon hasonló a lineáris keresés 2.8 programozási tételéhez. Az eltéréseket az alábbi táblázattal foglalhatjuk össze:

feladat		lineáris keresés 2.8
k	$\leftrightarrow$	i
$lnko(p, i) = 1$	$\leftrightarrow$	$\beta(i)$

Ez a visszavezetés *paraméteres visszavezetés*, mert az állapottér bővebb, és a helyettesítő táblázatban a $\beta(i)$ helyettesítésekor fel is használjuk ezt a plusz komponenst. Ugyanakkor a bevezetett p az előfeltétel szerint adott értékű és a program során nem változik (az utófeltételben is szerepel rejtve a $p = p'$).

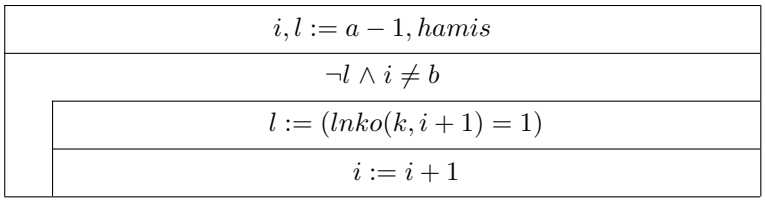
$k, l := m - 1, hamis$
$\neg l \wedge k \neq n$
$l := (lnko(p, k+1) = 1)$
$k := k + 1$

13.6. példa: Állapítsuk meg, hogy van-e az f függvény értékei között olyan szám, amely k-hoz relatív prím!
Megoldás:
$f : [m, n] \rightarrow \mathbb{N}$
$A = \begin{array}{ccccccc} \mathbb{Z} & \times & \mathbb{Z} & \times & \mathbb{L} & \times & \mathbb{N} \\ a & & b & & l & & k \end{array}$
$B = \begin{array}{ccccc} \mathbb{Z} & \times & \mathbb{Z} & \times & \mathbb{Z} \\ a' & & b' & & k' \end{array}$
$Q : (a = a' \wedge b = b' \wedge a \leq b + 1 \wedge k = k')$
$R : (Q \wedge l = (\exists j \in [a..b] : lnko(k, j) = 1))$

Ezt a lineáris keresés 2.8 tételre vezethetjük vissza, azonban a visszavezetés általánosított alteres, hiszen nem vagyunk kíváncsiak a keresés által visszaadott függvényargumentumra, amellyel a függvényérték relatív prím k-hoz. Ugyanakkor a visszavezetés paraméteres is k paraméter szerint.

A helyettesítések:

feladat		lineáris keresés 2.8
a	$\leftrightarrow$	m
b	$\leftrightarrow$	n
$lnko(k,i)=1$	$\leftrightarrow$	$\beta(i)$

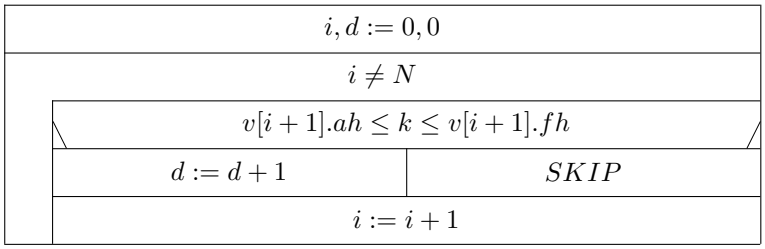


13.7. példa: Adott a kezdőpontja szerint növekvő sorrendben a számegyenes N darab intervalluma. Állapítsuk meg, hogy a k szám hány intervallumba esik bele.
Megoldás:
$Iv = (ah : \mathbb{Z}, fh : \mathbb{Z})$
$I_{Iv}(i) = (i.ah \leq i.fh + 1)$
$V = vekt([1..N], Iv)$
$A = \underset{v}{V} \times \underset{k}{\mathbb{Z}} \times \underset{d}{\mathbb{N}_0}$
$B = \underset{v'}{V} \times \underset{k'}{\mathbb{Z}}$
$Q : (v = v' \wedge k = k' \wedge \forall i \in [1..N-1] : v[i].ah \leq v[i+1].ah)$
$R : (Q \wedge d = \sum_{i=1}^{N} \chi(v[i].ah \leq k \leq v[i].fh))$

A specifikáció nagyon hasonló a számlálás programozási tételéhez. Az eltéréseket az alábbi táblázattal foglalhatjuk össze:

feladat		számlálás
1	$\leftrightarrow$	m
N	$\leftrightarrow$	n
$v[x].ah \leq k \leq v[x].fh$	$\leftrightarrow$	$\beta(x)$
i	$\leftrightarrow$	k

A visszavezetés paraméteres a k szerint, továbbá a számlálás programozási tétele nem követeli meg a bemenő vektor valamilyen rendezettségét, tehát a feladat előfeltétele erősebb, mint a tételé, így szigorítunk.



13.8. példa: Döntsük el a monoton növekedő f függvényről a szigorú értelemben vett monotonitást!
Megoldás:

$f : [m..n] \to \mathbb{Z}$
$A = \underset{m}{\mathbb{Z}} \times \underset{n}{\mathbb{Z}} \times \underset{l}{\mathbb{Z}}$
$B = \underset{m'}{\mathbb{Z}} \times \underset{n'}{\mathbb{Z}}$
$Q : (m = m' \wedge n = n' \wedge m \leq n + 1 \wedge \forall i \in [m..n-1] : f(i) \leq f(i+1))$
$R : (Q \wedge l = (\forall i \in [m..n-1] : f(i) < f(i+1))) = (Q \wedge \neg l = (\exists i \in [m..n-1] : f(i) \geq f(i+1)))$

Ha az utófeltétel második alakját vesszük figyelembe, akkor a specifikáció hasonlít a lineáris keresés 2.8 specifikációjához, pár apró eltéréstől eltekintve. Ilyen eltérés (a táblázatban összefoglalhatókon kívül), hogy a tétel i állapottér-komponense a mi specifikációnkban nem szerepel, illetve az, hogy a tétel nem követelné meg a vizsgált függvény monotonitását. Tehát egyrészről i szerint általánosított alteres, másrészről – az előfeltételek különbözősége miatt – szigorításos ez a visszavezetés.

feladat		lineáris keresés 2.8
$f(i) \geq f(i+1)$	$\leftrightarrow$	$\beta(i)$
$\neg l$	$\leftrightarrow$	l
$n-1$	$\leftrightarrow$	n

Az n-nek $n-1$-gyel való helyettesítése esetén az előfeltétel csak akkor teljesül, ha $m \leq n$, ezért a visszavezetéshez erre a kiegészítő feltételre is szükség van.

A visszavezetés paraméteres a k szerint, továbbá a számlálás programozási tétele nem követeli meg a bemenő vektor valamilyen rendezettségét, tehát a feladat előfeltétele erősebb, mint a tételé, így a visszavezetés egyben szigorított is.

$i, \neg l := m-1, hamis$	
$\neg\neg l \wedge i \neq n-1$	
	$\neg l := f(i+1) \geq f(i+2)$
	$i := i+1$

Ezzel a struktogrammal több probléma is van. A legnagyobb az, hogy szerepelnek benne nem definiált értékadások. Hiszen az értékadás definíciójakor a jelölésben csak azt engedtük meg, hogy a bal oldalon változónevek álljanak. Itt azonban egy kifejezés, nevezetesen a $\neg l$ áll, ahol l már változónév. Egyszerű meggondolások alapján azonban láthatjuk, hogy az ilyen típusú értékadások átírhatók érvényessé, ha „mindkét oldalt még egyszer tagadjuk”, majd a kialakult $\neg\neg l$ formulák helyére egyszerűen csak l-et írunk. Ezzel az utolsó lépéssel a struktogram másik fölösleges alakú kifejezését, a benne szereplő $\neg\neg l$ részt is kiküszöböltük.

Így:

$i, l := m-1, igaz$	
$l \wedge i \neq n-1$	
	$l := f(i+1) < f(i+2)$
	$i := i+1$

13.9. példa: Keressünk az $[a..b]$ intervallumban ikerprímeket! Feltehetjük, hogy $a > 2$.

Megoldás:
$A = \begin{array}{cccccccc} \mathbb{N} & \times & \mathbb{N} & \times & \mathbb{L} & \times & \mathbb{N} \\ a & & b & & l & & p \end{array}$
$B = \begin{array}{ccc} \mathbb{Z} & \times & \mathbb{Z} \\ a' & & b' \end{array}$
$Q : (a = a' \wedge b = b' \wedge a \leq b + 1 \wedge a > 2)$
$R : (Q \wedge l = (\exists i \in [a..b-2] : (\text{prím}(i) \wedge \text{prím}(i+2))) \wedge l \rightarrow (p \in [a..b-2] \wedge \text{prím}(p) \wedge \text{prím}(p+2)))$

A specifikáció hasonlít a lineáris keresés 2.8 specifikációjához, pár apró eltéréstől eltekintve. Ezeket foglalja össze a táblázat:

feladat		lineáris keresés 2.8
$\text{prím}(i) \wedge \text{prím}(i+2)$	$\leftrightarrow$	$\beta(i)$
p	$\leftrightarrow$	i
a	$\leftrightarrow$	m
$b-2$	$\leftrightarrow$	n

Az utolsó helyettesítés esetében az előfeltétel csak akkor igaz, ha $a \leq b - 1$, ezért ezt fel kell tennünk.

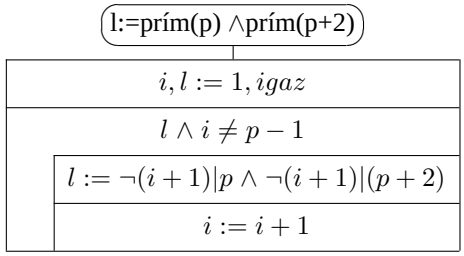
A visszavezetés szigorított, hiszen az előfeltétel az intervallumok tekintetében szigorúbb a feladatban, míg az utófeltétel gyengébb a feladatban, mint a tételben (mi nem követeljük meg, hogy az első ikerprímet találja meg a program).

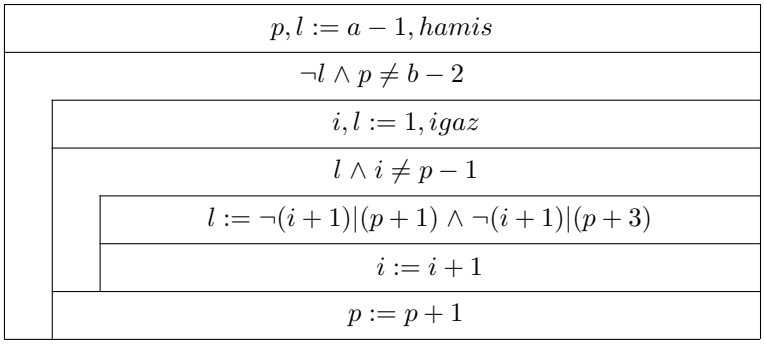
$p, l := a - 1, hamis$	
$\neg l \wedge p \neq b - 2$	
	$l := \text{prím}(p+1) \wedge \text{prím}(p+3)$
	$p := p + 1$

A megoldásban szereplő $l := \text{prím}(p+1) \wedge \text{prím}(p+3)$ értékadást egy új feladatként is megfogalmazhatjuk: állapítsuk meg egy p számról, hogy ő maga és a nála kettővel nagyobb szám prím-e! Ez a feladat az eredeti állapottér egy alterén specifikálható:
$A' = \begin{array}{cccc} \mathbb{L} & \times & \mathbb{N} \\ l & & p \end{array}$
$B' = \begin{array}{c} \mathbb{N} \\ p' \end{array}$
$Q' : (p = p' \wedge p > 2)$
$R' : (Q \wedge l = (\forall i \in [2..p-1] : (\neg(i|p) \wedge \neg(i|(p+2))))) = (Q \wedge \neg l = (\exists i \in [2..p-1] : (i|p \vee i|(p+2))))$

Ez visszavezethető a lineáris keresés 2.8-ra. A visszavezetés az alteres visszavezetés mindkét esetével összeegyeztethető (hiszen az intervallum kezdetét konstanssal helyettesítettük, ugyanakkor a tétel i eredménykomponensét nem használtuk a specifikációban). A visszavezetés általános is, mivel mi a 2 konstanssal mint az intervallum elejével és a $p - 1 \geq 2$ kifejezéssel mint az intervallum végével az üres intervallum feldolgozását kizártuk az előfeltételben, így az szigorúbb a tétel előfeltételénél.

feladat		lineáris keresés 2.8
$i\|p \vee i\|(p+2)$	$\leftrightarrow$	$\beta(i)$
$\neg l$	$\leftrightarrow$	l
2	$\leftrightarrow$	m
$p-1$	$\leftrightarrow$	n





13.10. példa: Keressük meg az $f : \mathbb{Z} \to \mathbb{Z}$ függvény értékei között a k szám p-edik előfordulását az $[a..b]$ intervallumban!

Megoldás:

A feladat megfogalmazásához definiálunk egy $g \in \mathbb{Z} \to \mathbb{N}_0$ parciális függvényt, ahol $g(i)$ azt adja meg, hogy hányszor vette fel az f függvény a k értéket az $[a..i]$ intervallumban.

$g(a-1) = 0, \forall i \in [a-1, b-1]$:

$$g(i+1) = \begin{cases} g(i), & \text{ha } f(i+1) \neq k; \\ g(i)+1, & \text{ha } f(i+1) = k. \end{cases}$$

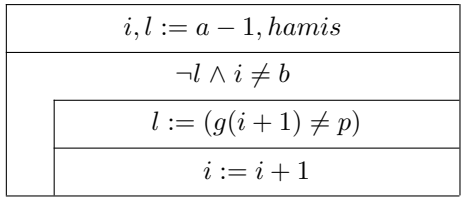
$$\begin{array}{lccccccccccc} A = & \mathbb{Z} & \times & \mathbb{Z} & \times & \mathbb{Z} & \times & \mathbb{N} & \times & \mathbb{Z} & \times & \mathbb{L} \\ & a & & b & & k & & p & & i & & l \end{array}$$

$$\begin{array}{lccccccc} B = & \mathbb{Z} & \times & \mathbb{Z} & \times & \mathbb{Z} & \times & \mathbb{N} \\ & a' & & b' & & k' & & p' \end{array}$$

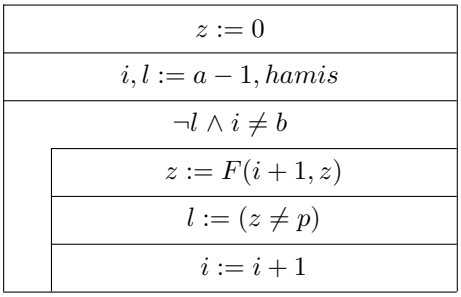
$Q : (a = a' \wedge b = b' \wedge a \leq b + 1 \wedge k = k' \wedge p = p')$

$R : (Q \wedge l = (\exists j \in [a..b] : g(j) = p) \wedge l \to (i \in [a..b] \wedge g(i) = p \wedge f(i) = k))$

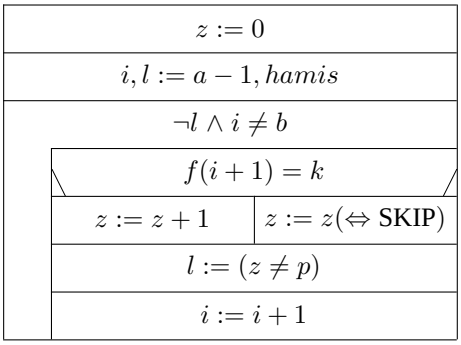
feladat		lineáris keresés 2.8
a	$\leftrightarrow$	m
b	$\leftrightarrow$	n
$g(i) = p$	$\leftrightarrow$	$\beta(i)$



Felhasználva a rekurzív függvény változóval való helyettesítésének programtranszformációs módszerét g-re:



Ha ismerjük az elágazással definiált függvény kiszámításának programozási tételét, akkor ebből megkaphatjuk a megoldóprogramot, ha azt F-re alkalmazzuk:



13.11. példa: Keressük meg az $[m..n]$ intervallumban $f : \mathbb{Z} \to \mathbb{Z}$ függvény egy olyan értékét, amely egyenlő a közvetlen szomszédai átlagával!

Megoldás:

$A = \underset{m}{\mathbb{Z}} \times \underset{n}{\mathbb{Z}} \times \underset{l}{\mathbb{L}} \times \underset{i}{\mathbb{Z}}$

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

$Q = (m = m' \wedge n = n' \wedge m \leq n + 1)$

$R = (Q \wedge l = (\exists j \in [m+1, n-1] : (f(j) = \frac{f(j-1)+f(j+1)}{2}) \wedge l \rightarrow (i \in [m+1, n-1] \wedge f(i) = \frac{f(i-1)+f(i+1)}{2}))$

A lineáris keresés 2.8 specifikációja hasonló. Lássuk, hogy az alábbi átnevezés után milyen különbségek maradnak:

feladat		lineáris keresés 2.8
$m+1$	$\leftrightarrow$	m
$n-1$	$\leftrightarrow$	n
$f(i) = \frac{f(i-1)+f(i+1)}{2}$	$\leftrightarrow$	$\beta(i)$

Az első két helyettesítés esetében a tétel előfeltétele csak akkor igaz, ha $m < n$, ezért ezt fel kell tennünk.

A visszavezetés általánosított, hiszen nem követeljük meg, hogy az első ilyen speciális tulajdonságú elemet találjuk meg.

$i, l := m, hamis$
$\neg l \wedge i \neq n-1$
$l := (f(i+1) = \frac{f(i)+f(i+2)}{2})$
$i := i+1$

13.8. Feladatok

A feladatokban szereplő függvények (ha más nincs kikötve) egész számok egy intervallumán vannak értelmezve és egész értékűek.

13.1. Határozzuk meg az f függvény azon pozitív értékeinek a számát, amelyek közvetlenül egy negatív érték után állnak!

13.2. Állapítsuk meg, hogy az n természetes számnak van-e páratlan valódi osztója!

13.3. Határozzuk meg az n természetes szám legkisebb páratlan osztóját!

13.4. Adottak az x és y vektorok, ahol y elemei az x indexei közül valók. Keressük meg az x vektornak az y-ban megjelölt elemei közül a legnagyobbat!

13.5. Adjuk meg, hány olyan elem van az x vektorban, amely kisebb az indexénél!

13.6. Határozzuk meg az n természetes szám legkisebb egyszeres osztóját!

13.7. Adottak az azonos értelmezési tartományú f és g függvények. Az f értékei egészek, g pedig csak a $0, 1$ értékeket veszi fel. Határozzuk meg azoknak a páros f-értékeknek a számát, amelyek olyan pozícióban vannak, ahol a g függvény értéke 1!

13.8. Keressük meg az f függvény értékei között azt a számot, amelynek decimális alakjában az egyesek helyén a legnagyobb számjegy áll!

13.9. Az x vektor egy szöveget tartalmaz. Állapítsuk meg, hogy visszafelé olvasva a szöveg ugyanaz-e!

13.10. Adott egy gráf a csúcsmátrixával. Állapítsuk meg a k-adik csúcs fokszámát!

13.11. Határozzuk meg az f függvény legnagyobb k-val osztható értékét!

13.12. Határozzuk meg az n természetes szám valódi páros osztóinak számát!

13.13. Határozzuk meg az f függvény lokális minimumai közül a legnagyobbat! (Egy érték akkor lokális minimum, ha mindkét szomszédjánál kisebb.)

13.14. Adjuk meg az f függvény egy k-val osztható értékéhez tartozó argumentumát!

13.15. Határozzuk meg az f függvénynek a k-nál kisebb legnagyobb értékét!

13.16. Adott a középpontjával és a sugarával a síkon egy kör, és további N darab pont. Keressünk egy olyan pontot, amely a körbe esik!

13.17. Határozzuk meg az f függvénynek az $[a, b]$ intervallumba eső legnagyobb értékét!

13.18. Határozzuk meg az f függvény azon értékeinek a számát, amelyek vagy az $[a, b]$ vagy a $[c, d]$ intervallumba esnek!

13.19. Határozzuk meg az f függvénynek azt a legnagyobb értékét, amely k-val osztva 1-et ad maradékul!

13.20. Adjuk meg az f függvénynek azt az értékét, amely $mod\ N$ a legnagyobb!

13.21. Adottak az azonos értelmezési tartományú f és g függvények. Az $(f(i), g(i))$ számpárok egy-egy síkbeli pont koordinátái. Számoljuk meg, hogy a pontok közül hány esik az (x_0, y_0) középpontú, r sugarú körbe!

13.22. Keressük meg az f függvénynek az első n-nél kisebb vagy 0 értékét!

13.23. Adottak az x és y vektorok, valamint a k szám. Az y vektor az x indexeinek egy részhalmazát tartalmazza. Számoljuk meg, hány olyan k-val osztható elem van x-ben, amelynek indexe megtalálható y-ban!

13.24. Adott az x vektorban egy szöveg. Állapítsuk meg, hogy a szöveg tartalmaz-e magánhangzót!

13.25. Keressük meg az f függvény egy olyan értékét, amely beleesik az $[a, b]$ és a $[c, d]$ intervallumba is!

13.26. Az x vektor egy szöveget tartalmaz. Számoljuk meg, hány magánhangzó van a szövegben!

13.27. Állapítsuk meg, hol van a monoton növekedő f függvényben a legnagyobb ugrás, azaz az $f(k) - f(k-1)$ érték mely k-ra maximális!

13.28. Határozzuk meg az f függvény legnagyobb páros értékéhez tartozó argumentumot!

13.29. Keressünk az x vektorban két olyan szomszédos elemet, amelyek szorzata negatív!

13.30. Adottak az $[m, n]$ intervallumon értelmezett f és g függvények. Állapítsuk meg, hány egész koordinátájú pont esik a függvényértékek közé!

13.31. Adottak az x és b vektorok. „Fektessük” b-t az x vektorra folyamatosan egymás után, ahányszor csak lehet, és számoljuk meg, hány helyen egyeznek az egymás feletti értékek!

13.32. Adott az n természetes szám. Határozzuk meg n egy valódi osztóját!

13.33. Adjuk meg az f függvénynek azt az értékét, amelynek szomszédai átlagától való eltérése a legnagyobb!

13.34. Keressünk az x vektorban egy olyan elemet, amely osztható az indexével!

13.35. Adott az x mátrix, amelynek elemei sorfolytonosan növekvő sorozatot alkotnak. Keressük meg a mátrixban az n értéket!

13.36. Adott egy x vektor, amely színeket tartalmaz sötétedő sorrendben (színeken értelmezve van egy úgynevezett sötétségi reláció, amely teljes rendezés). Keressük meg az x vektorban a világoskéket!

13.37. Adott a síkon N darab pont. Keressük meg az origótól legtávolabb eső pontot!

13.38. Adjuk meg az f függvény utolsó pozitív értékének argumentumát!

13.39. Adott az x vektor, amelynek elemei nullák és egyesek. Számoljuk meg, hányszor fordul elő a vektorban a '0101' szakasz!

13.40. Állapítsuk meg, hogy van-e az f függvény értékei között olyan szám, amely k-hoz relatív prím!

13.41. Határozzuk meg az n természetes szám legkisebb valódi nem prím osztóját!

13.42. Adottak az f és g monoton növő függvények, valamint a k szám. Állapítsuk meg, található-e olyan i és j argumentum, amelyre $f(i) + g(j) = k$!

13.43. Adjuk meg az x mátrix egy olyan sorának indexét, amely nem tartalmaz pozitív elemet!

13.44. Állapítsuk meg, hogy a b vektorban levő szöveg előfordul-e a karakteres x vektorban!

13.45. Adott az injektív f függvény. Adjuk meg az f egy olyan értékét, amelyet legalább egy nála nagyobb megelőz!

13.46. Határozzuk meg az f függvénynek azt az értékét, amely leghamarabb fordul elő másodszor!

13.47. Keressük meg az x mátrixnak azt a sorát, amelynek minden eleme 1!

13.48. Adjuk meg, hány prímszám van az $[a, b]$ intervallumban!

13.49. Határozzuk meg az n-nél kisebb, n-hez relatív prím természetes számok számát!

13.50. Adott két egybevágó $2n$-szög. Mindkettő oldalait véletlenszerűen kékre vagy pirosra festettük. Helyezzük egymásra a két sokszöget úgy, hogy a lehető legtöbb helyen legyenek azonos színű oldalak egymáson!

13.51. Számoljuk meg az $f : [m, n] \times [m, n] \to \mathbb{Z}$ függvény nulla értékeit!

13.52. Számoljuk meg, hogy az x mátrixban hány olyan sor van, amely csak egyetlen nullától különböző elemet tartalmaz!

13.53. Állapítsuk meg, melyik az f függvény leggyakrabban felvett értéke!

13.54. Határozzuk meg az f függvénynek azt az értékét, amelyet a legtöbb nála nagyobb elem előz meg!

13.55. Keressük meg a négyzetes x mátrixnak azt az oszlopát, amelyben a főátlóbeli feletti elemek összege a legnagyobb!

13.56. Határozzuk meg a négyzetes x mátrixnak a főátlóbeli alatti legnagyobb elemét!

13.57. Számoljuk meg, hogy az x mátrixnak hány olyan sora van, amely csak egy nullától különböző elemet tartalmaz!

13.58. Adott a sík N pontja. Állapítsuk meg, melyik a két legtávolabbi pont!

13.59. Egy sakkbajnokság végeredményét egy x négyzetes mátrixban tároltuk, ahol az i-edik sorban a j-edik elem az i-edik játékos és a j-edik játékos közti mérkőzés eredményét jelenti az i-edik játékos szempontjából ($x_{i,j}$ értéke 0, ha j nyert; 2, ha i nyert; 1, ha a játékosok döntetlenben egyeztek meg). Keressük meg a bajnokság (egyik) győztesét (győztes az, akinek a legtöbb pontja van)!

13.60. Adott egy $f : [a, b] \to \mathbb{Z}$ függvény. E függvény értelmezési tartományának egy i pontját a függvény csúcsának nevezzük, ha $\forall j \in [i + 1, b] : f(j) < f(i)$. Adjuk meg, hány csúcsa van f-nek!

13.61. Keressük meg az x négyzetes mátrixnak azt a fődiagonálissal párhuzamos átlóját, amelyben az elemek összege a legnagyobb!

13.62. Adott a $0, 1$ értékeket felvevő f függvény. Keressük meg a függvény értelmezési tartományának azt az elemét, amely a leghosszabb egyes-értéksorozat kezdete!

13.63. Egy függvény értelmezési tartományának azt a szakaszát, amelyhez tartozó értékek negatívok úgy, hogy a szakaszt jobbról és balról nemnegatív érték vagy az értelmezési tartomány vége határolja, a függvény negatív szigetének nevezzük. Adjuk meg az f függvény értelmezési tartományában a negatív szigetek számát!

13.64. Állapítsuk meg, van-e negatív szám az f függvényértékeinek (kezdő) részletösszegei között!

13.65. Adjunk meg egy olyan k számot, amelyre az n természetes szám bináris alakjának k-adik helyi értékén 1-es áll!

13.66. Adjuk meg az f függvény értelmezési tartományának azt a leghosszabb szakaszát, amelyen belül az értékek növekvőek!

13.67. Állapítsuk meg, hogy az x szám binárisan felírt alakjában hány darab 1-es szerepel!

13.68. Adott az x vektor, amelynek elemei karakterek. A vektor szavakat tartalmaz, amiket egy-egy vessző választ el egymástól. Adjuk meg a leghosszabb szónak a kezdőindexét!

13.69. Egy $f : [a, b] \to \mathbb{Z}$ függvény értelmezési tartományának azon szakaszát, melynek két végpontja a függvény lokális minimumhelye úgy, hogy a végpontok közötti elemek nem azok, a függvény egy hegyének nevezzük. Adjuk meg a legszélesebb hegy kezdőpontját!

13.70. Egy vektornak azt a szakaszát, amely csupa negatív elemet tartalmaz úgy, hogy a szakaszt jobbról és balról nemnegatív elem vagy a vektor vége határolja, a vektor negatív szigetének nevezzük. Adjuk meg az x vektor legnagyobb negatív szigetének kezdőindexét!

13.71. Egy múzeumban az i-dik órában x_i látogató érkezik és y_i látogató megy el. Melyik órában volt a legtöbb látogató a múzeumban?

13.72. Adott az egész számok egy vektora és két egész szám. Állapítsuk meg, hogy a két adott szám előfordul-e a vektorban; és ha igen, akkor melyik előbb!

13.73. Helyezzünk el n darab vezért egy $n \times n$ méretű sakktáblán úgy, hogy egyik vezér se támadjon másikat!

13.74. Hányféleképpen helyezhetünk el n darab vezért egy $n \times n$ méretű sakktáblán úgy, hogy egyik vezér se támadjon másikat?

13.75. Legfeljebb hány vezért lehet egy $n \times n$ méretű tórikus sakktáblán elhelyezni úgy, hogy egyik vezér se támadjon másikat?

13.76. Adott n fiú és ugyanennyi lány. Egy x logikai mátrixban tároljuk a fiúk és lányok közötti szimpátiát (ez egy szimmetrikus reláció) a következőképpen: $x_{i,j}$ igaz, ha az i-edik fiú és a j-edik lány szimpatizál egymással, hamis, ha nem szimpatizálnak egymással. A feladat az, hogy ha lehet, akkor párosítsuk (házasítsuk) össze őket úgy, hogy minden párban a felek szimpatizáljanak!

13.77. Adott egy $n \times m$ méretű sakktábla (i, j) mezején egy huszár. Végig lehet-e vezetni a táblán úgy, hogy minden mezőre lép, de csak egyszer, és minden lépése szabályos (huszárlépés)?

13.78. Hányféleképpen lehet n forintot kifizetni m különböző címletű pénzből?

13.79. Hányféleképpen lehet n forintot kifizetni m különböző címletű pénzből, ha legfeljebb $d_1, d_2, \ldots, d_m$ használható fel?

13.80. Adott a természetes számok egy S véges részhalmaza. Kiválasztható-e ebből n darab elem úgy, hogy az összegük m legyen?

13.81. Az x szekvenciális fájl (megengedett művelet az $sx, dx, x : read$) egy vállalat dolgozóiról a következő adatokat tartalmazza:

- a dolgozó azonosító száma;
- vezető beosztásban van-e;
- legmagasabb iskolai végzettsége.

Válasszuk ki a v sorozatba azoknak a dolgozóknak az adatait, akik vezető beosztásban vannak, a z sorozatba azoknak az azonosítóit, akik vezető beosztásban vannak és nem érettségiztek!

13.82. Az x szekvenciális fájl (megengedett művelet az $sx, dx, x : read$) földrengések adatait tartalmazza. Egy elem a következőkből áll:

- az észlelés helyének koordinátái;
- a rengés erőssége;
- a rengés időtartama;
- a földrész azonosítója;
- a rengést előre jelezték-e.

Válasszuk ki a t sorozatba az előre nem jelzett földrengések észlelési helyeit, a z sorozatba pedig a 20 másodpercnél hosszabb földrengések adatait!

13.83. Adott a keresztnevek és a virágnevek fájlja, mindkettő ábécésorrendben rendezett (megengedett művelet az $sx, dx, x : read$). Határozzuk meg azokat a keresztneveket, amelyek nem virágnevek!

13.84. Adott egy egész számokat tartalmazó fájl. Ha a fájl tartalmaz pozitív elemet, akkor keressük meg a legnagyobbat, különben a legkisebbet!

13.85. Egy fájlban (megengedett művelet a $lopop$ extremális elemmel) adottak az egyes kaktuszfajtákról a (név, őshaza, virágszín, méret) adatok. Válogassuk ki egy fájlba a mexikói, egy másikba a piros virágú kaktuszokat!

13.86. Adott egy fájlban (megengedett művelet az $sx, dx, x : read$) egy OTP-nyilvántartás (név, összeg) párok alakjában. Adjuk meg annak a nevét, akinek nincs tartozása, de a legkisebb a betétállománya (ha van ilyen)!

13.87. Az x szekvenciális fájl egész számokat tartalmaz (megengedett művelet az $sx, dx, x : read$). Keressünk a fájlban lokális maximumot, vagyis olyan értéket, amely mindkét közvetlen szomszédjánál nagyobb!

13.88. Adott az x vektor és az y szekvenciális fájl, amelynek elemei egyaránt pozitív egész számok. x-ben és y-ban egy szám legfeljebb egyszer fordul elő, és mindkettő növekvően rendezett. Az y egy olyan szekvenciális fájl, amelyre csak a $lopop$ művelet megengedett, és a fájl végét egy negatív szám jelzi. Állítsuk elő a rendezett z sorozatot, ami x és y elemeit tartalmazza!

13.89. Adottak az x szekvenciális fájlban (megengedett művelet az $sx, dx, x : read$) egy évfolyam hallgatóinak adatai. Egy elem a hallgató nevét, csoportszámát és tíz db osztályzatot (a nulla azt jelöli, hogy az osztályzat hiányzik) tartalmaz. A fájl a csoportszámok szerint növekvően rendezett. Állítsuk elő az y sorozatot, ami a hallgatók nevét, csoportszámát és átlagát tartalmazza!

14. fejezet

Absztrakciós stratégia

A programozási feladatok megoldása során mindig valamilyen absztrakciós eljárást követünk, ami a megoldás során lehet végig azonos, de lépésenként változhat is. A különböző programozási módszereket az jellemzi, hogy döntően milyen absztrakciós stratégiára építenek.

Az előző részben mutattunk egy egyszerű példát az állapotér-transzformációra. Az állapottér-transzformációra épülő stratégiákat *adatabsztrakciós* stratégiának is nevezzűk, hiszen az állapotér az adatok absztrakciója.

Most ezt a feladatot újra megoldjuk, de egészen más gondolatmenettel. Nem az állapoteret alakítjuk át, hanem az eredeti állapottéren definiálunk olyan függvényeket, amelyek segítségével az utófeltétel kényelmesen felírható, és könnyebben kezelhető feladathoz jutunk. Ebben az esetben *függvényabsztrakcióról* beszélünk.

Tehát a feladat a következő:Tegyük fel, hogy van egy karakterekből álló szekvenciális fájlunk, ami egy szöveget tartalmaz. Számoljuk meg, hogy hány olyan csupa betükből álló rész van a szövegben, amelynek hossza nagyobb, mint egy megadott k érték!

Legyen x egy karakterekből álló sorozat. Definiáljuk a következő parciális függvényt:

$f(0) = 0$ és

$\forall i \in [0..dom(x) - 1]$:

$$f(i+1) = \begin{cases} f(i)+1, & \text{ha } x_{i+1} \in BET\ddot{U}; \\ 0, & \text{ha } x_{i+1} \notin BET\ddot{U}. \end{cases}$$

Az f függvény azt adja meg, hogy az i helyen hanyadik betüje van egy megszakítás nélküli betüsorozatnak az x-ben. Ezért a feladat specifikációja:

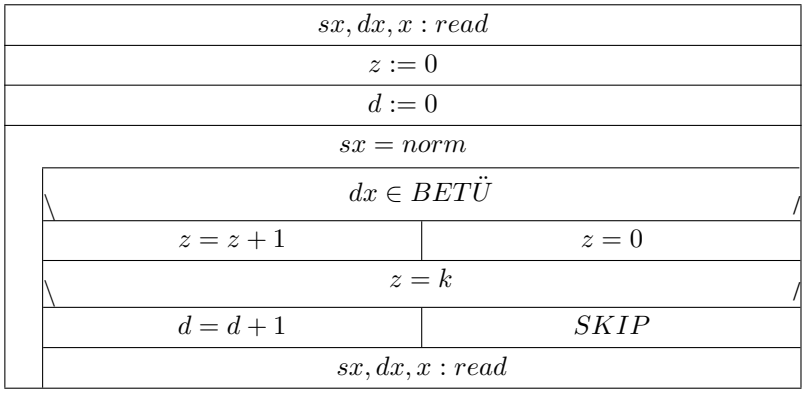
$A = \underset{x}{X} \times \underset{d}{\mathbb{N}}$ ha $X = file(CH)$,

$B = \underset{x'}{x}$

$Q : (x = x')$

$R : (k = k' \wedge d = \sum_{i=0}^{dom(x')} \chi(f(i) = k))$

Erre a feladatra alkalmazva a megszámlálás tételének szekvenciális fájlra vonatkozó változatát és a rekurzív függvény helyettesítése változóval transzformációt kapjuk a következő megoldó programot.



A következőkben egy összetettebb feladaton mutatjuk meg e két alapvető stratégia összehasonlítását.

14.1. Az időszerűsítés definíciója

Először néhány, a sorozatok (szevenciális fájlok) estén gyakran használt jelölést, fogalmat vezetünkbe. Legyen x egy sorozat. $\{x\}$ jelöli a sorozat elemeinek halmazát, azaz

$$\{x\} = \bigcup_{i=1}^{dom(x)} \{x_i\}.$$

Gyakran használunk olyan sorozatot, amelynek *kulcsa* van. Ez azt jelenti, hogy $S = seq(X)$, ahol $X = (k : K, d : D)$ és K egy rendezett halmaz. K a kulcsok, D az adatrészek lehetséges értékeinek halmaza. Legyen $s \in S$, ekkor $\{s.k\}$ az s kulcsainak halmaza, azaz

$$\{s.k\} = \bigcup_{i=1}^{dom(s)} \{s_i.k\}.$$

Azt mondjuk, hogy az $s \in S$ sorozat *kulcs szerint rendezett*, ha

$$\forall i \in [1..dom(s) - 1] : s_i.k \leq s_{i+1}.k.$$

Ha S minden eleme kulcs szerint rendezett, akkor S kulcs szerint rendezett sorozat típus.

Ha az $s \in S$ sorozat minden elemének kulcsa különbözik, azaz

$$\forall i, j \in [1..dom(s)], i \neq j : s_i.k \neq s_j.k,$$

akkor azt mondjuk, hogy s *egyértelmü kulcsú* sorozat. Ha S minden eleme egyértelmü kulcsú, akkor S egyértelmü kulcsú sorozat típus.

Legyen S egyértelmü kulcsú sorozat típus. Definiáljuk a következő $\mathcal{K} :\in S \times K \rightarrow X$ parciális függvényt:

$$(s, q) \in \mathcal{D}_{\mathcal{K}} \Leftrightarrow q \in \{s.k\}$$

és

$$\mathcal{K}(s, q) \in \{s\} \text{ és } \mathcal{K}(s, q).k = q.$$

Most lássuk az időszerűsítés feladatát. Induljunk ki egy olyan adatfájlból, amelyben azonos típusú elemek találhatóak. Ezt a fájlt *törzsfájl*nak fogjuk nevezni. Legyen az elemek típusa E, ekkor

$T = seq(E)$.
Legyen adott továbbá egy olyan fájl, amely transzformációk sorozatát tartalmazza. Ezt a fájlt fogjuk *módosítófájl*nak nevezni. Legyen $F = \{f \mid f : T \to T\}$, ekkor
$M = seq(F)$.
A feladat az, hogy időszerűsítsük a törzsfájlt a módosítófájlban leírt transzformációkkal. Jelölje Υ az időszerűsítés transzformációt. Ekkor $\Upsilon : T \times M \to T$ és

$$\Upsilon(t, m) = m_{dom(m)} \circ \cdots \circ m_2 \circ m_1(t).$$

Ahhoz, hogy a feladatra megoldást adjunk, ez a leírás még túl általános, ezért további kikötéseket teszünk a fájlokra.

1. Az időszerűsítés kulcsos.

   Legyen $E = (k : K, d : D)$ és $F = (k : K, v : V)$, ahol K kulcs része; D a törzsrekord adat része; V pedig az elvégzendő transzformációt definiáló típus. Feltesszük még, hogy mind a tözsfájl, mind a módosítófájl a kulcs (k) szerint rendezett.

2. A törzsfájl a kulcs szerint egyértelmű.

3. A transzformáció csak a következő háromféle (törlés, beszúrás, javítás) lehet:

$$\begin{aligned} V &= (tr : W_1; be : W_2; jav : W_3); \\ W_1 &= \{\alpha\}, \\ W_2 &= (d : D), \\ W_3 &= (g : G), \quad \text{ahol } G = seq(H), H = \{\gamma \mid \gamma : D \to D\}. \end{aligned}$$

   Hátra van még annak a leírása, hogy hogyan hatnak a fenti transzformációk a törzsfájlra. Mivel a törzsfájl kulcs szerint egyértelmű és rendezett, elég csak azt megadni, hogy milyen elemekből áll.

   Három esetet különböztetünk meg attól függően, hogy a transzformáció törlés, beszúrás vagy javítás.

   (a) $m_i.tr$, azaz m_i törlés:

$$\{m_i(t)\} = \begin{cases} \{e \mid e \in t \text{ és } e.k \neq m_i.k\}, & \text{ha } m_i.k \in \{t.k\}; \\ \{t\} & \text{különben.} \end{cases}$$

   (b) $m_i.be$, azaz m_i beszúrás:

$$\{m_i, t\} = \begin{cases} \{t\} \cup \{(m_i.k, m_i.v.d)\}, & \text{ha } m_i.k \notin \{t.k\}; \\ \{t\} & \text{különben.} \end{cases}$$

   (c) $m_i.jav$, azaz m_i javítás:

$$\{m_i, t\} = \begin{cases} \{e \mid e \in t \text{ és } e.k \neq m_i.k\} \cup \\ \{(m_i.k, m_i.g_{dom(m_i.g)} \circ \ldots & \text{ha } m_i.k \in \{t.k\} \\ \cdots \circ m_i.g_1(\mathcal{K}(t, m_i.k).d))\}; & \\ \{t\} & \text{különben.} \end{cases}$$

4. A javítás művelet csere.

   Ez azt jelenti, hogy $dom(m_i.g) = 1$ és $m_i.g_1$ konstans. Ebben az esetben a $m_i.g_1(\mathcal{K}(t,k).d) = m_i.d$.

A feladat specifikációja tehát:

$$A = \underset{t_0}{T} \times \underset{m}{M} \times \underset{t}{T}$$

$$B = \underset{t_0'}{T} \times \underset{m'}{M}$$

$Q : (t_0 = t_0' \wedge m = m'$ és az $1..x$ pontok teljesülnek$)$

$R : (t = \Upsilon(t_0', m'))$

Ha a fenti specifikációban $x = 3$, akkor *közönséges*, ha $x = 4$, akkor *egyszerű* időszerűsítésről beszélünk.

14.2. Időszerűsítés egyértelmű módosítófájllal

A továbbiakban az egyszerű időszerűsítéssel fogunk foglalkozni. Közönséges időszerűsítés esetére csak utalni fogunk.

A feladatot három irányból is megpróbáljuk megoldani: visszavezetjük halmazok uniójára, egyváltozós egyértékű; illetve kétváltozós egyértékű elemenkénti feldolgozásra.

14.2.1. Visszavezetés halmazok uniójára

Ez a megoldás Dijkstra kőnyvében [Dij 76] található. Az alapgondolat az, hogy a t elemei vagy olyanok, hogy a kulcsuk t_0-ban is kulcs vagy olyanok, hogy a kulcsuk m-ben is kulcs. Tehát $\{t\} = x \cup y$, ahol $x = \{e \in \{t\} \mid e.k \in \{t_0.k\}$ és $y = \{e \in \{t\} \mid e.k \in \{m.k\}$.

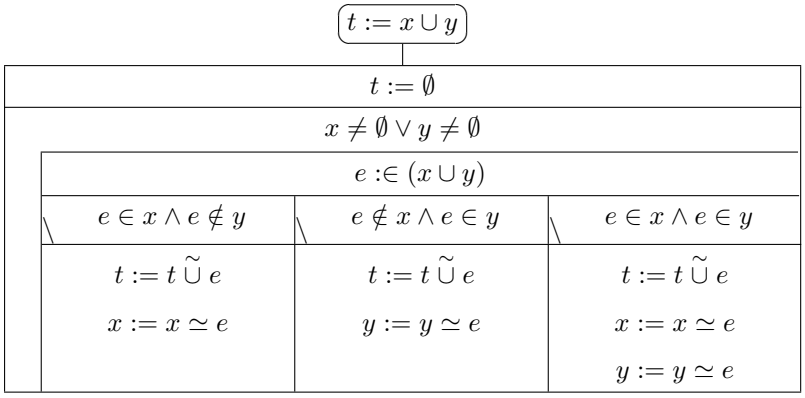
A transzformált feladat tehát a következő: $A = \underset{x}{T} \times \underset{y}{T} \times \underset{t}{T}$

$$B = \underset{x'}{T} \times \underset{y'}{T}$$

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

$R : (t = x \cup y)$

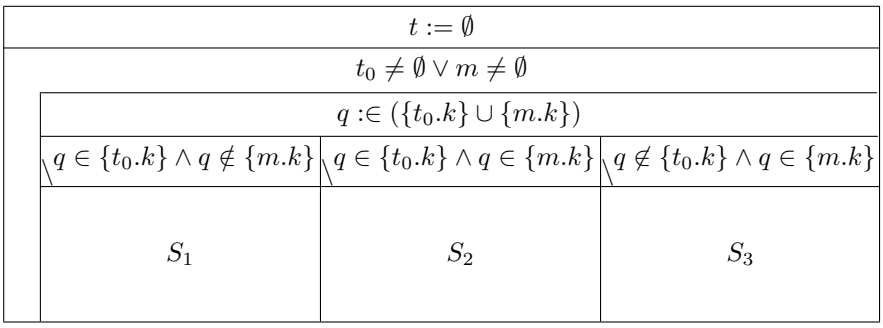
Idézzük fel az *unió* programját:



Most már csak a fenti programot kell transzformálni az eredeti állapottérre.

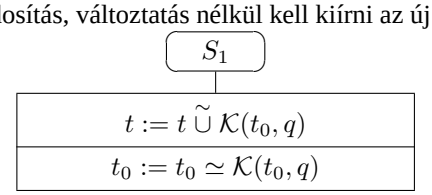
$$\begin{aligned} x \neq \emptyset \vee y \neq \emptyset &\rightarrow t_0 \neq \emptyset \vee m \neq \emptyset \\ e :\in (x \cup y) &\rightarrow e.k :\in (\{t_0.k\} \cup \{m.k\}) \\ e \in x &\rightarrow e.k \in \{t_0.k\} \\ e \in y &\rightarrow e.k \in \{m.k\} \end{aligned}$$

Jelöljük q-val az $e.k$ kulcsot, ekkor a megfelelő program:

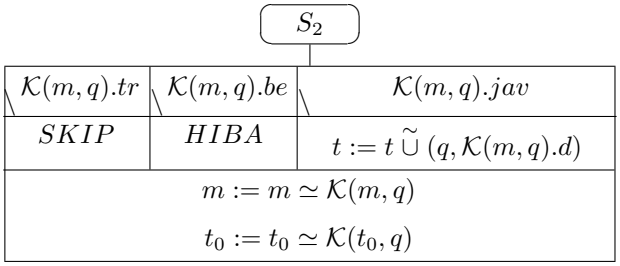


Vizsgáljuk meg most, hogy mit kell tenni az elágazás egyes ágaiban:

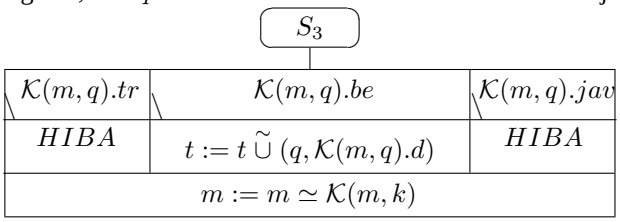
1. Ha a q kulcs csak a t_0 eredeti törzsfájlban szerepelt, akkor a $\mathcal{K}(t, q)$ elemre nem vonatkozott módosítás, változtatás nélkül kell kiírni az új törzsfájlba.



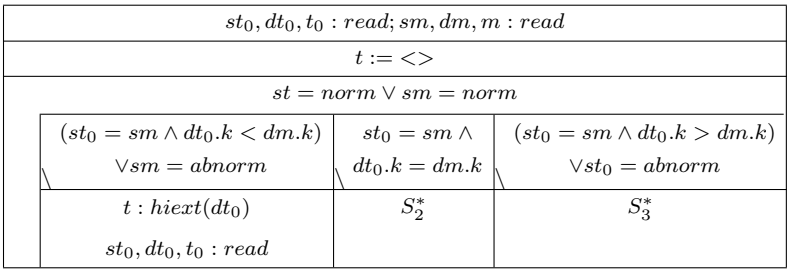
2. Ha a q kulcsérték mind az eredeti törzsfájlban, mind pedig a módosítófájlban szerepelt, akkor a törlés és a javítás műveletek végezhetők el. Ebben az ágban a q kulcsú elemet mindkét fájlból el kell hagyni.



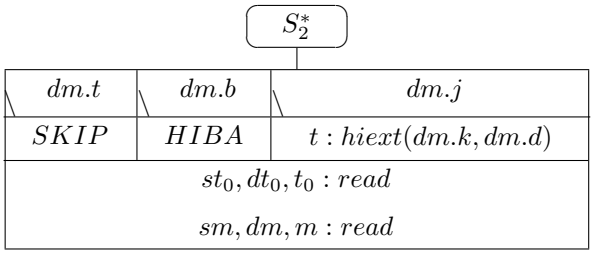
3. Ha a k kulcs csak a módosítófájlban szerepelt, akkor csak a beszúrás műveletet lehet elvégezni, és a q kulcsú elemet ki kell törölni a módosítófájlból.

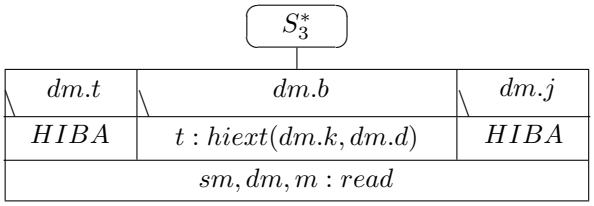


Ha a programnak hibajelzést is kell adnia, akkor azt a fenti struktogramokban $HIBA$-val jelzett helyeken kell megtennie. Térjünk most vissza az eredeti feladatra, ahol halmazok helyett szekvenciális fájlok szerepelnek. Legyen az input fájlokon a $read$ művelet értelmezve. Ekkor a program az alábbiak szerint alakul:



ahol





14.2.2. Visszavezetés egyváltozós elemenkénti feldolgozásra

Állapottér-transzformációt alkalmazunk. Legyen $X = seq(Y)$ és $Y = (k : K, d : D', v : V')$, ahol $D' = D \cup \{\text{"üres"}\}$ és $V' = V \cup \{\text{"üres"}\}$.

Legyen $x \in X$ és $\{x.k\} = \{t_0.k\} \cup \{m.k\}$.

$$\forall i \in [1..x.dom] : x_i.d = \begin{cases} \text{"üres"}, & \text{ha } x_i.k \notin \{t_0.k\}; \\ \mathcal{K}(t_0, x_i.k).d, & \text{ha } x_i.k \in \{t_0.k\}. \end{cases}$$

$$x_i.v = \begin{cases} \text{"üres"}, & \text{ha } x_i.k \notin \{m.k\}; \\ \mathcal{K}(m, x_i.k).v, & \text{ha } x_i.k \in \{m.k\}. \end{cases}$$

Legyen $f : X \to T$.

$$f(x) = \bigcup_{e \in \{x\}} f(\{e\}) \text{ és}$$

$$f(\{e\}) = \begin{cases} (e.k, e.d), & \text{ha } e.v = \text{"üres"}; \\ \\ \text{"üres"}, & \text{ha } e.v.tr \wedge e.d = \text{"üres"}, \\ (e.k, e.v.d), & \text{ha } e.v.be \wedge e.d = \text{"üres"}, \\ \text{"üres"}, & \text{ha } e.v.jav \wedge e.d = \text{"üres"}; \\ \\ \text{"üres"}, & \text{ha } e.v.tr \wedge e.d \neq \text{"üres"}, \\ (e.k, e.d), & \text{ha } e.v.be \wedge e.d \neq \text{"üres"}, \\ (e.k, e.v.d), & \text{ha } e.v.jav \wedge e.d \neq \text{"üres"}. \end{cases}$$

A specifikáció:

$$A = \underset{x}{X} \times \underset{t}{T}$$

$$B = \underset{x'}{X}$$

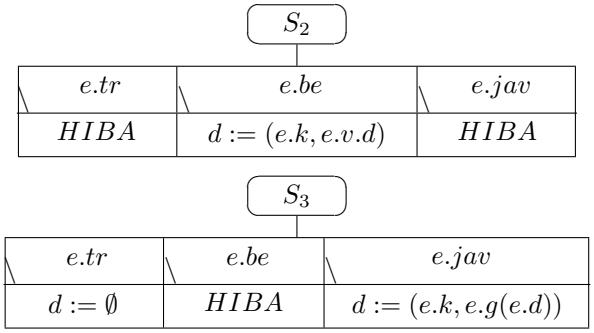
$$Q : (x = x')$$

$$R : (t = f(x'))$$

A halmazokra felírt megoldóprogram:

<table>
<tr><td colspan="4">$t := \emptyset$</td></tr>
<tr><td rowspan="5"></td><td colspan="3">$x \neq \emptyset$</td></tr>
<tr><td colspan="3">$e :\in x$</td></tr>
<tr><td>$e.d \neq \text{"üres"} \wedge$
$e.v = \text{"üres"}$</td><td>$e.d \neq \text{"üres"} \wedge$
$e.v = \text{"üres"}$</td><td>$e.d = \text{"üres"} \wedge$
$e.v = \text{"üres"}$</td></tr>
<tr><td>$d := (e.k, e.d)$</td><td>S_2</td><td>S_3</td></tr>
<tr><td colspan="3">$t := t \mathbin{\widetilde{\cup}} d$</td></tr>
<tr><td colspan="4">$x := x \simeq e$</td></tr>
</table>

ahol



Térjünk vissza most az eredeti állapottérre:

$x \neq \emptyset$	$\Rightarrow$	$t_0 \neq \emptyset \vee m \neq \emptyset$
$e :\in x$	$\Rightarrow$	$q :\in (\{t_0.k\} \cup \{m.k\})$
$e.d \neq$ "üres" $\wedge\, e.v =$ "üres"	$\Rightarrow$	$q \in \{t_0.k\} \wedge q \notin \{m.k\}$
$e.d =$ "üres" $\wedge\, e.v \neq$ "üres"	$\Rightarrow$	$q \notin \{t_0.k\} \wedge q \in \{m.k\}$
$e.d \neq$ "üres" $\wedge\, e.v \neq$ "üres"	$\Rightarrow$	$q \in \{t_0.k\} \wedge q \in \{m.k\}$
$x := x \simeq e$	$\Rightarrow$	$q \in \{t_0.k\} \wedge q \notin \{m.k\}$: $t_0 := t_0 \simeq \mathcal{K}(t_0, q)$ $q \in \{t_0.k\} \wedge q \in \{m.k\}$: $t_0 := t_0 \simeq \mathcal{K}(t_0, q)$; $m := m \simeq \mathcal{K}(m, q)$ $q \notin \{t_0.k\} \wedge q \in \{m.k\}$: $m := m \simeq \mathcal{K}(m, q)$

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 $t := t \,\widetilde{\cup}\, d$ értékadást csak azokba az ágakba írjuk bele, amelyekben $d \neq \emptyset$. Ekkor ugyanazt a programot kapjuk, mint az első megoldásban.

14.2.3. Visszavezetés kétváltozós elemenkénti feldolgozásra

A feladat megoldásának talán legegyszerűbb módja az, ha kétváltozós egyértékű – hibakezelés esetén kétértékű – elemenkénti feldolgozásra vezetjük vissza, úgy, hogy az elemeket a kulcsukkal azonosítjuk az elemeket. Tekintsük a feladat eredeti specifikációját.

Ha a módosítófájl kulcs szerint egyértelmű, akkor az időszerűsítés függvénye (Υ) a kulcsokra nézve elemenként feldolgozható. A kulcsértékekre felírt függvény:

$$\Upsilon(q, \emptyset) = \mathcal{K}(t_0, q).$$

$$\Upsilon(\emptyset, q) = \begin{cases} \emptyset, & \text{ha } \mathcal{K}(m,q).tr; \\ (q, \mathcal{K}(m,q).d), & \text{ha } \mathcal{K}(m,q).be; \\ \emptyset, & \text{ha } \mathcal{K}(m,q).jav. \end{cases}$$

$$\Upsilon(q, q) = \begin{cases} \emptyset, & \text{ha } \mathcal{K}(m,q).tr; \\ \mathcal{K}(t_0, q), & \text{ha } \mathcal{K}(m,q).be; \\ (q, \mathcal{K}(m,q).g(\mathcal{K}(t_0,q).d)), & \text{ha } \mathcal{K}(m,q).jav. \end{cases}$$

Ha ezt a fenti függvényt behelyettesítjük a kétváltozós elemenkénti feldolgozás tételébe, akkor ugyanahhoz a megoldóprogramhoz jutunk – csak sokkal rövidebb úton –, mint az első megoldásban.

A három megoldást összehasonlítva, két észrevételt tehetünk. Meglepő módon három látszólag teljesen különböző megoldási stratégiával ugyanahhoz a programhoz jutottunk. Valójában az eredmény nem annyira meglepő, hiszen tudjuk, hogy az unió kétváltozós elemenkénti feldolgozás.

A másik észrevételünk az, hogy minél erősebb tételt alkalmazunk, annál egyszerűbben jutunk el a megoldóprogramhoz.

14.3. Időszerűsítés nem egyértelmű módosítófájllal

Vajon miben változik a feladat, ha a módosítófájl kulcs szerint nem egyértelmű? Ebben az esetben a feladat nem elemenként feldolgozható, hiszen az elemek nem tekinthetők azonosnak a kulcsukkal. Erre a problémára kétféle megoldási módot is megvizsgálunk: az egyik jellemzően *adatabsztrakciós* a másik elsődlegesen *függvényabsztrakciós* megközelítés.

14.3.1. Megoldás adatabsztrakcióval

Mint az előbb már említettük, ha a módosító fájl kulcs szerint nem egyértelmű, akkor a feladat nem elemenként feldolgozható. Próbáljuk meg azzá tenni. Ehhez arra van szükség, hogy a módosítófájlt kulcs szerint egyértelművé tegyük. Ezt egy állapottér-transzformáció segítségével könnyen elérhetjük, ugyanis csak annyit kell tennünk, hogy az azonos kulcsú módosítórekordokat egy új rekordba fogjuk össze. Így az új módosítórekord a következőképpen fog kinézni:

(kulcs, transzformációsorozat)

Az állapottér transzformációt két lépésben adjuk meg. Legyen $V = seg(U)$, ahol $U = seq(F)$.

$$\begin{array}{lccccc} A' = & T & \times & V & \times & T \\ & t_0 & & v & & t \end{array}$$

A v-re teljesül, hogy $\forall i \in [1..dom(v)] : \forall j \in [1..dom(v_i)-1] : v_{i_j}.K = v_{i_{j-1}}.K$, v egyértelmű kulcsú, és a kapcsolat m és v között: $seq(m|F) = seq(v|F)$.

Legyen $S = seq(V)$; és $F' = (k : K, s : S)$ és $X = seq(F')$.

$$\begin{array}{lccccc} A' = & T & \times & X & \times & T \\ & t_0 & & x & & t \end{array}$$

x kulcs szerint egyértelmű, $\{x.K\} = \{v.K\}$ és $seq(x|V) = seq(v|V)$.

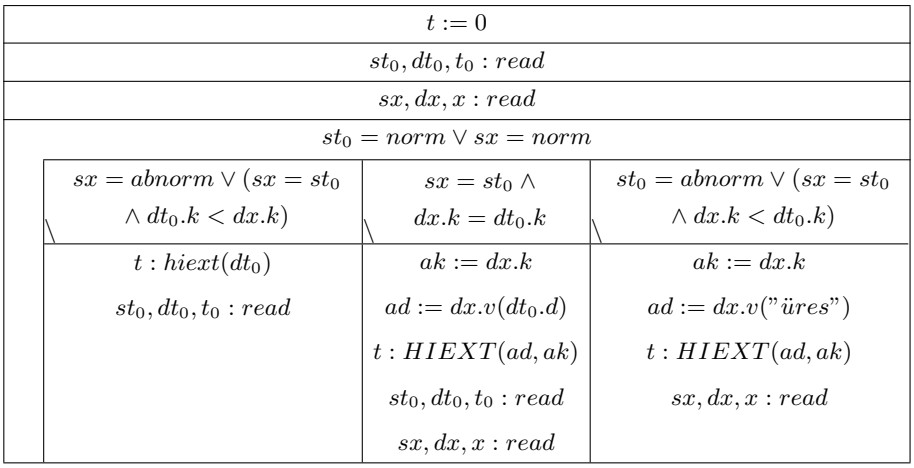
Ezzel a módosítófájllal tehát eljutottunk a kétváltozós egyértékű (hibafájl használata esetén kétértékű) elemenkénti feldolgozáshoz. Az egyetlen különbség csak az, hogy az adott transzformációsorozat végrehajtását meg kell valósítanunk az eredeti állapottéren. Ehhez szükségünk lesz az "$\ddot{u}res$" szimbólumra, amely értéket hozzávesszük az adat rész típusához: $D' = D \cup \{"\ddot{u}res"\}$. Az, hogy egy törzsrekord adatrésze "$\ddot{u}res$", azt jelenti, hogy a rekordot nem kell kiírni az eredményfájlba.

Az elemenként feldolgozható függvényünk:

$$\begin{aligned} \Upsilon(e, \emptyset) &= \mathcal{K}(t_0, e.k) \\ \Upsilon(\emptyset, e) &= (e.k, \mathcal{K}(x, e.k).s("\ddot{u}res")) \\ \Upsilon(e, e) &= (e.k, \mathcal{K}(x, e.k).s(\mathcal{K}(t_0, e.k).d)) \end{aligned}$$

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 Υ függvény egyértelműen definiált.

Írjuk fel tehát a fenti megfontolások alapján a kétváltozós elemenkénti feldolgozás programját a megfelelő behelyettesítésekkel:



Definiálnunk kell még, hogy mit jelent a fenti struktogramban a transzformációsorozat elvégzése. Ehhez bevezetjük az $f : \mathbb{N}_0 \to D'$ rekurzívan definiált függvényt:

$$dx.v(p) = f(dx.v.dom),$$

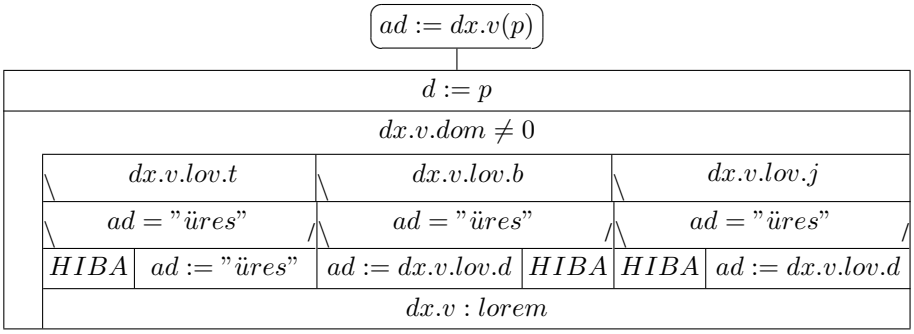
ahol

$$\begin{aligned} f(0) &= p; \\ f(i+1) &= dx.v_{i+1}(f(i)). \end{aligned}$$

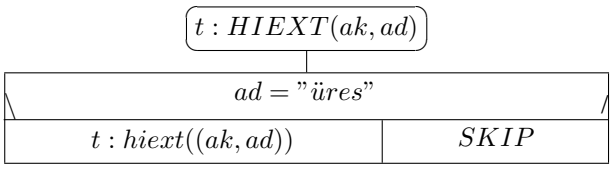
A fenti definícióban szereplő $dx.v_{i+1}$ művelet elvégzésén az alábbi függvényértékeket értjük. Legyen $d \in D'$, ekkor:

$$dx.v_{i+1}(d) = \begin{cases} \text{"}\ddot{u}res\text{"}, & \text{ha } dx.v.t \wedge d \neq \text{"}\ddot{u}res\text{"}; \\ d, & \text{ha } dx.v.t \wedge d = \text{"}\ddot{u}res\text{"}; \\ d, & \text{ha } dx.v.b \wedge d \neq \text{"}\ddot{u}res\text{"}; \\ dx.v.d, & \text{ha } dx.v.b \wedge d = \text{"}\ddot{u}res\text{"}; \\ dx.v.d, & \text{ha } dx.v.j \wedge d \neq \text{"}\ddot{u}res\text{"}; \\ d, & \text{ha } dx.v.j \wedge d = \text{"}\ddot{u}res\text{"}. \end{cases}$$

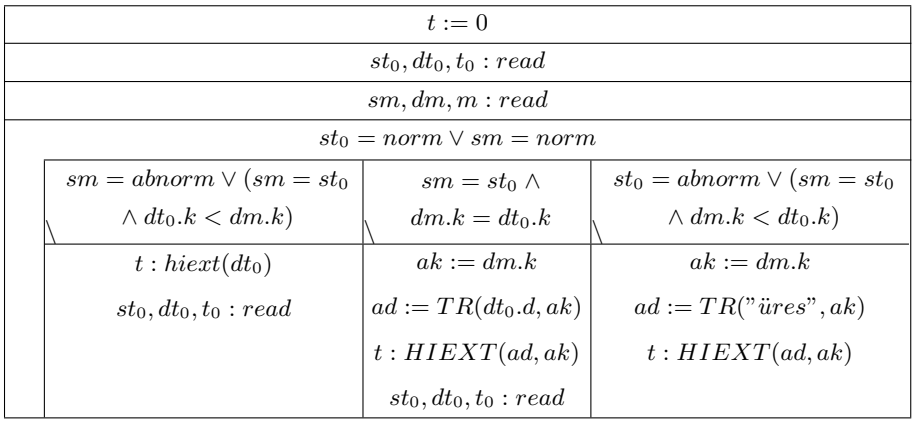
Az f függvényt kiszámító – a módosításssorozatot elvégző – program:



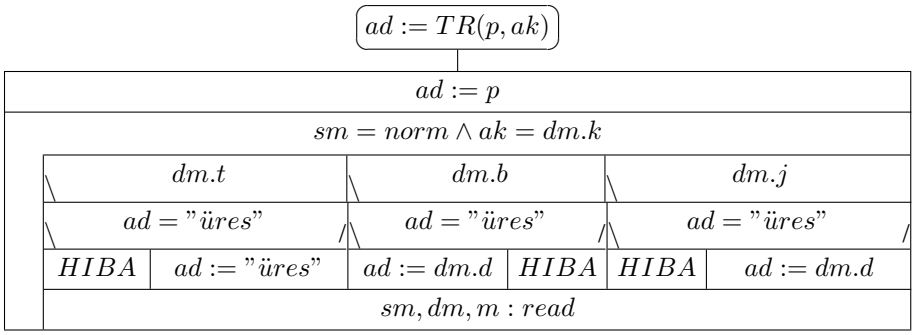
Mivel az aktuális adat értéke lehet "$\ddot{u}res$" is, a $hiext$ művelet helyett az alábbi programot használjuk:



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:



14.3.2. 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.

Kulcsok egyértelműsítése. A gyakorlatban sokszor találkozhatunk olyan fájlokkal, amelyek rekordjaiban van valamilyen kulcsmező, ami szerint a fájl rendezett, de a fájl mégsem egyértelmű kulcs szerint és így a kulcsmezőre vonatkoztatva nem elemenként feldolgozható. A következőkben egy olyan technikát fogunk bemutatni, amelyben egy új kulcsot definiálunk a fájlra, és ezen új kulcs szerint a fájl már egyértelmű.

Tegyük fel, hogy a fájl $U = (k : K, z : Z)$ 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 $g : seq(U) \to seq(V)$:

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.

Megoldás extremális elemmel. Induljunk ki egy olyan absztrakt fájlból, mint amilyet az egyváltozós elemenkénti feldolgozásra való visszavezetésben használtunk. Természetesen, mivel most a módosítófájl nem egyértelmű kulcs szerint, az X absztrakt fájl definíciója kissé módosul: ha egy kulcs mindkét fájlban 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 adat része "$\ddot{u}res$".

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

$$\{g(x).h\} = \{g(t_0).h\} \cup \{g(m).h\} \cup \{(extr, 1)\},$$

és $\forall i \in [1..x.dom]$:

$$g(x)_i.d = \begin{cases} \mathcal{K}(g(t_0), g(x)_i.h).d, & \text{ha } g(x)_i.h \in \{g(t_0).h\}; \\ \text{"}\ddot{u}res\text{"} & \text{különben.} \end{cases}$$

$$g(x)_i.v = \begin{cases} \mathcal{K}(g(m), g(x)_i.h).v, & \text{ha } g(x)_i.h \in \{g(m).h\}; \\ \text{"}\ddot{u}res\text{"} & \text{különben.} \end{cases}$$

Figeljük meg, hogy ez a megoldás egy a gyakorlatban sokszor hasznos eszközt használ, az utolsó utáni elem bevezetését ($read$ extremális elemmel).

A jobb áttekinthetőség érdekében a függvényt most komponensenként fogjuk felírni. $f : \mathbb{N}_0 \to T \times K' \times D'$, $f = (f_1, f_2, f_3)$:

$$f(0) = (<>, <\ddot{u}res>, <\ddot{u}res>)$$

$$f_1(i+1) = \begin{cases} f_1(i), & \text{ha } f_2(i) = x_{i+1}.k \vee \\ & \quad (f_2(i) \neq x_{i+1}.k \wedge f_3(i) = \text{"}\ddot{u}res\text{"}); \\ hiext(f_1(i), (f_2(i), f_3(i))), & \text{ha } f_2(i) \neq x_{i+1}.k \wedge f_3(i) \neq \text{"}\ddot{u}res\text{"}. \end{cases}$$

$$f_3(i+1) = \begin{cases} x_{i+1}.v(f_3(i)), & \text{ha } f_2(i) = x_{i+1}.k; \\ x_{i+1}.v(x_{i+1}.d), & \text{ha } f_2(i) \neq x_{i+1}.k. \end{cases}$$

$$f_2(i+1) = \begin{cases} f_2(i), & \text{ha } f_2(i) = x_{i+1}.k; \\ x_{i+1}.k, & \text{ha } f_2(i) \neq x_{i+1}.k. \end{cases}$$

Ezt a függvényt használva a feladat specifikációja:

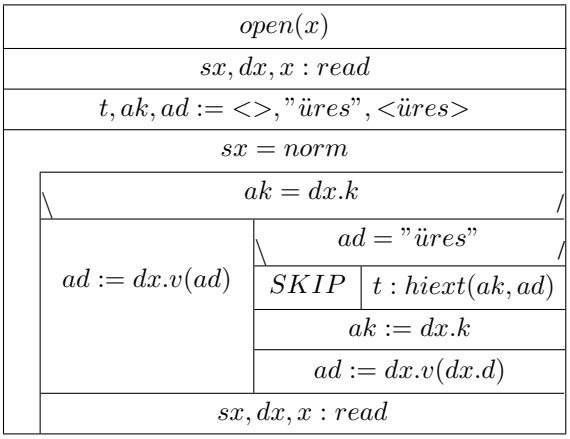
$$A = \underset{x}{X} \times \underset{t}{T}$$

$$B = \underset{x'}{X}$$

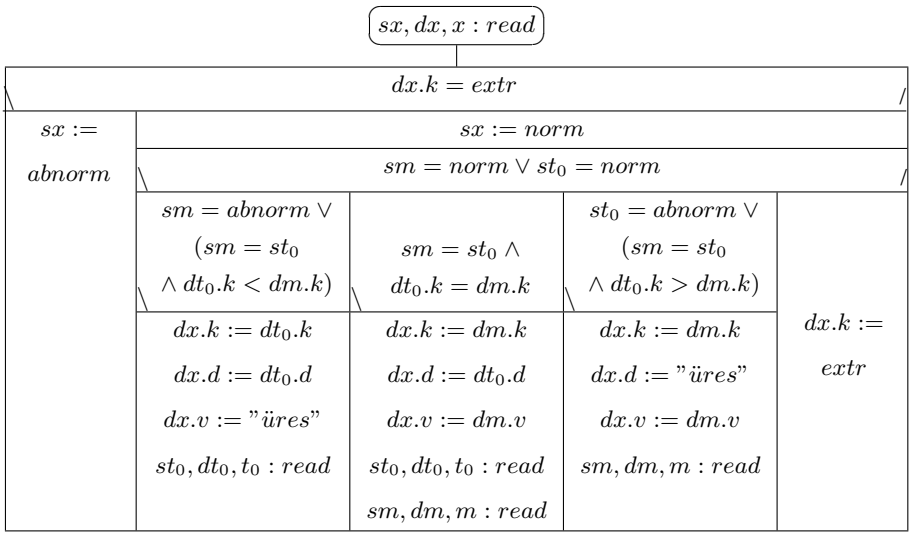
$Q : (x = x')$

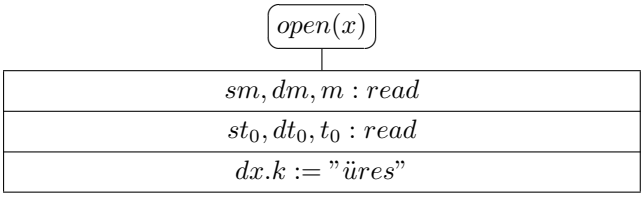
$R : (t = f_1(x'.dom))$

Ez a feladat visszavezethető a fájlra felírt rekurzívan megadott függvény helyettesítési értékének kiszámítására:

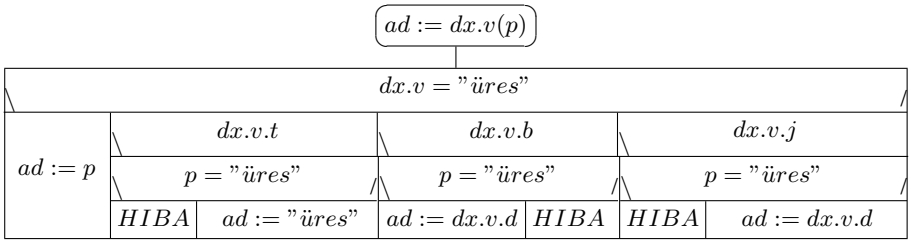


Az x absztrakt fájl műveleteinek megvalósítása:





A transzformáció elvégzésének megvalósítására az x absztrakt fájlt 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.

Megoldás függvénykompozícióval. Olyan megoldást is adhatunk a feladatra, amelynek a specifikációjában az utófeltétel egy kompozícióval adott függvény kiszámítása, ahol a kompozíció egy rekurzív módon megadott függvényből és egy esetszétválasztással definiált függvényből áll. Ebben az esetben nincs szükség utolsó utáni elemre.

$$A = \underset{x}{X} \times \underset{t}{T}$$
$$B = \underset{x'}{X}$$
$$Q : (x = x')$$
$$R : (t = HIEXT(f_1(x'.dom), (f_2(x'.dom), f_3(x'.dom)))),$$

ahol $f : \mathbb{N}_0 \to T \times K \times D'$,

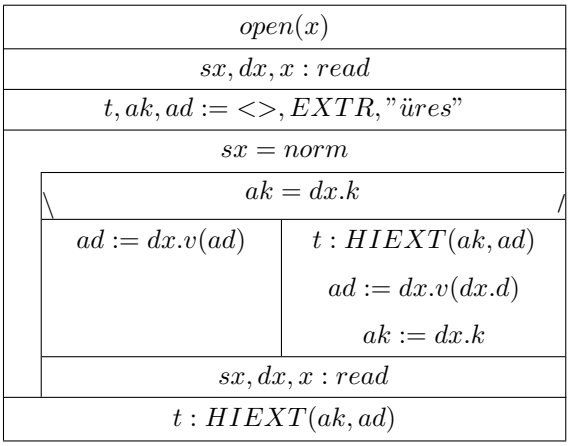
$$f(0) = (<>, EXTR, \text{"üres"})$$
$$f(i+1) = \begin{cases} (f_1(i), f_2(i), x_{i+1}.v(f_3(i))), & \text{ha } x_{i+1}.k = f_2(i); \\ (HIEXT(f_1(i), (f_2(i), f_3(i))), \\ \quad x_{i+1}.k, x_{i+1}.v(x_{i+1}.d)), & \text{ha } x_{i+1}.k \neq f_2(i); \end{cases}$$

és $HIEXT : T \times (K \times D') \to T$,

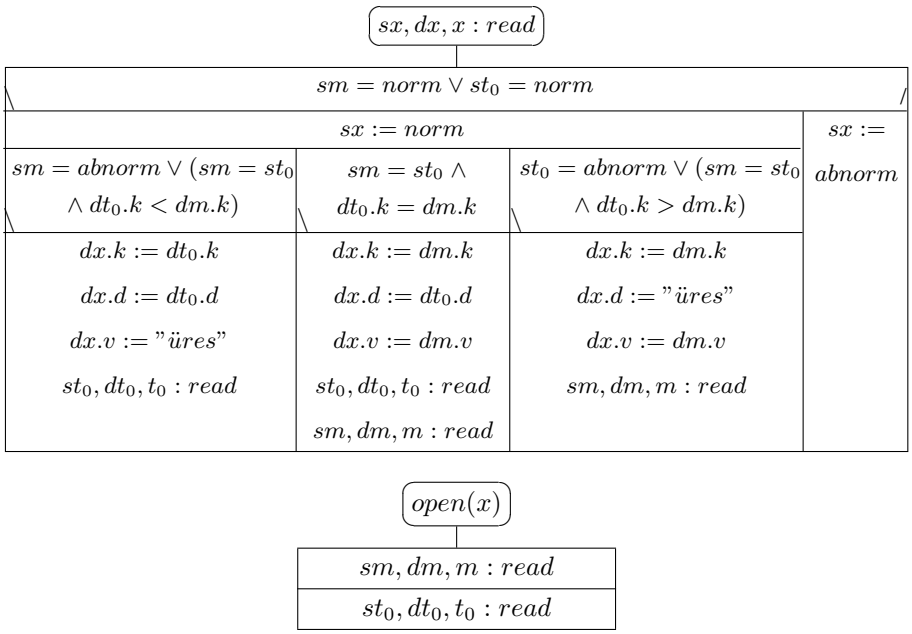
$$HIEXT(t, k, d) = \begin{cases} hiext(t, (k, d)), & \text{ha } d \neq \text{"üres"}; \\ t, & \text{ha } d = \text{"üres"}. \end{cases}$$

Az f függvény kezdőértékének definíciójában szereplő $EXTR$ kulcsérték tetszőleges olyan kulcsérték lehet, amely egyik fájlban sem fordul elő.

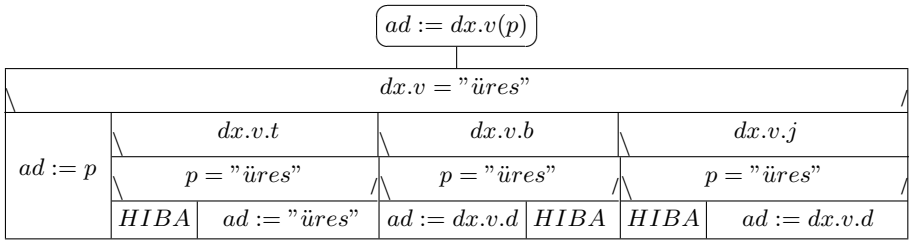
Mivel az utófeltétel függvénykompozícióval adott, a megoldás egy szekvencia lesz, amelynek első része az f, második része pedig a $HIEXT$ függvényt számítja ki. Az f függvény egyes komponenseinek rendre a t, ak, ad változók felelnek meg.



Az x absztrakt fájl műveleteinek megvalósítása:



Hátravan még a transzformáció elvégzésének megvalósítása:



14.4. Feladatok

14.1. Adott az egész számokat tartalmazó x vektor. Válogassuk ki az y sorozatba a vektor pozitív elemeit!

14.2. Adott két vektorban egy angol–latin szótár: az egyik vektor i-edik eleme tartalmazza a másik vektor i-edik elemének a jelentését. Válasszuk ki egy vektorba azokat az angol szavakat, amelyek szóalakja nem egyezik meg a latin megfelelőjével.

14.3. Adott egy x sorozat, ami egy vállalat dolgozóinak adataiból áll. Egy dolgozóról a következő adatokat tudjuk:

- azonosító szám;
- születési adatok (idő, hely, anyja neve);
- lakcím;
- iskolai végzettség;
- a munkaviszony kezdete;
- beosztás;
- fizetés.

Adott még az y sorozat, amely azonosítókat tartalmaz. Mindkét sorozat az azonosító szám szerint rendezett. Adjuk meg a z sorozatban azoknak a dolgozóknak az adatait, akiknek az azonosítója szerepel y-ban, és a munkaviszonyuk kezdete egy adott évnél régebbi!

14.4. Az x sorozat egy szöveget tartalmaz. Tömörítsük a szöveget úgy, hogy mindenütt, ahol több szóköz van egymás mellett, csak egy szóközt hagyjunk meg!

14.5. Adott egy szöveg, ami mondatokból áll, és a mondatok végén pont van. Módosítsuk a szöveget úgy, hogy minden mondat végét jelző pontot pontosvesszőre cserélünk! A mondatokban lehetnek idézetek, és az idézetek is tartalmazhatnak idézeteket tetszőleges mélységben (az idézetet egy kezdő idézőjel vezeti be és egy záró idézőjel jelzi a végét). Azok a pontok, amelyek egy idézet belsejében vannak, nem jelentik a mondat végét! Feltesszük, hogy a szövegben az idézőjelek kiegyensúlyozottak.

14.6. Adott az x sorozat, amely egy szöveget tartalmaz. Másoljuk át x-et a z sorozatba úgy, hogy a kerek zárójelek közé írt szöveget elhagyjuk! (A zárójelekkel együtt.)

14.7. Egy szekvenciális fájl (megengedett művelet az $sx, dx, x : read$) szöveget tartalmaz, melyben a szavakat szóközök (esetleg több szóköz) választják el. Számoljuk meg, hány 3 jelnél rövidebb szó van a szövegben!

14.8. Adott egy szekvenciális fájl (megengedett művelet az $sx, dx, x : read$), ami egy bank tranzakcióit tartalmazza: egy ügyfél adatait tartalmazó rekord után olyan rekordok következnek, amelyek az ügyfél tranzakcióit írják le.

- Ügyfél = (Azonosító, Számla összege)
- Tranzakció = (Kivét-betét, Összeg)

Állítsuk elő azt a fájlt, ami az ügyfeleknek a bankban levő pillanatnyi összegeit tartalmazza ügyfél típusú rekordokban!

14.9. Adott egy karakterekből álló szekvenciális fájl (megengedett művelet az $sx, dx, x : read$). Számoljuk meg, hogy a szöveg hány szóból áll, ha a 12 jelnél hosszabb szavakat két szónak tekintjük! (A szavakat tetszőleges számú szóköz választhatja el.)

14.10. Adott az x szekvenciális fájl (megengedett művelet az $sx, dx, x : read$), amely egy szöveget tartalmaz. Állapítsuk meg, hány olyan szó van a szövegben, amely tartalmaz „R” betűt!

14.11. Adott az x szekvenciális fájl, melynek elemei egy vezetéknevet és egy keresztnevet tartalmaznak. A fájl a keresztnevek szerint rendezett. Gyűjtsük ki a fájlból a különböző keresztneveket és azt, hogy hányszor szerepelnek!

14.12. Az x sorozat egy szöveget tartalmaz, ahol a szavakat egy vagy több szóköz választja el. Számoljuk meg, hogy a sorozatban hány k betűnél hosszabb szó van!

14.13. Adott egy évfolyam-nyilvántartás névsor szerint a következő adatokkal: név, csoportszám, átlag. Félévenként módosítják a nyilvántartást, ekkor az alábbi változások történhetnek:

- jöhetnek új hallgatók az évfolyamra;
- elmehetnek hallgatók (törölni kell a nyilvántartás-ból);
- változhat a hallgató átlaga;
- megváltozhat a hallgató csoportszáma.

A változások is névsor szerint rendezett fájlban vannak. Végezzük el az adatok frissítését.

14.14. Végezzük el egy áruház osztályain kint lévő árukészlet nyilvántartásának napi frissítését. Az adatbázisban az árukról nyilván van tartva, melyik osztályon árusítják, mi a neve, mennyi az ára, és hány db van belőle a boltban. A nyilvántartás osztályok szerint, azon belül árunév szerint rendezett. Egy módosítófájl tartalmazza ugyanígy rendezve a napi fogyást, egy másik az új szállítmányt, egy harmadik pedig az árváltozásokat.

14.15. Adott egy raktárnyilvántartás (áru, mennyiség) adatokkal, névsor szerint rendezve. Minden nap érkezik három fájl:

- a gyártótól: szállítás; melyik áruból mennyit hozott;
- a boltból: igénylés; melyik áruból mennyit rendel a bolt;
- a főnöktől: selejtezés; csak áruneveket tartalmaz, amiket törölni kell a nyilvántartásból.

Végezzük el az adatok módosítását. Ha nincs annyi áru, amennyit a bolt igényel, akkor ezt jelezzük, de amennyit lehet, adjunk ki a boltnak. Ha egy áru mennyisége 0-ra csökken, akkor töröljük ki a nyilvántartásból. Ha a gyártó olyan árut hoz, amilyen még nincs, azt fel kell venni a nyilvántartásba.

14.16. Adott a virágokról egy nyilvántartás (virágnév, szín, magasság) adatokkal, virágnév szerint növekvően rendezve egy x fájlban. Adott továbbá három ugyanilyen szerkezetű módosítófájl (a, b, c). Végezzük el az x fájl frissítését a következők szerint:

- Ha egy virág benne van a-ban, akkor szúrjuk be x-be.
- Ha egy virág benne van b-ben, akkor módosítsuk x-et a b-beli rekorddal.
- Ha egy virág benne van c-ben, akkor töröljük x-ből.

Ha egy rekord az a, b, c fájlok közül többen is benne van, akkor a módosításokat a fenti sorrendben kell elvégezni(először a, majd b és c).

Tárgymutató

Irodalomjegyzék

[Dij 68] Dijkstra, E. W.: Go to statement considered as harmful. *Comm. ACM.* 11, 3 (1968) 147–148.

[Mills 72] Mills, H. D.: Mathematical functions for structured programming, *SC 72-6012, IBM* Gaithersburg 1972.

[Hoa 72] Hoare, C. A.: Proof of correctness of data representations. *Acta informatica* 1 (1972) 271–281.

[Jac 75] Jackson, M. A.: *Principles of Programming Design*. Academic Press, 1975.

[Dij 76] Dijkstra, E. W.: *A Discipline of Programming*. Prentice-Hall, 1976.

[San 80] Sanderson, J. G.: *A Relational Theory of Computing*. Springer Verlag, 1980.

[Gri 81] Gries, D.: *The Science of Programming*. Springer Verlag, 1981.

[Fót 83] Fóthi Á.: *Bevezetés a programozáshoz*. Egyetemi jegyzet. ELTE TTK, Budapest, 1983.

[Mal 90] Malcolm, G.: Data Structures and Program Transformations. *Science of Computer Programming,* 14 (1990) 255–279.

[Bud 95] Budd, T.: *Multi-Paradigm Programming in Lede*. Addison-Wesley, 1995.

[GHJV 95] Gamma, E. – Helm, R. – Johnson, R. – Vlissides, J.: *Design Patterns – Elements of Reusable Object-Oriented Software*. Addision-Wesley,1995.

[Hor 95a] Horváth Z.: The Formal Specification of a Problem Solved by a Parallel Program – a Relational Model. *Annales Uni. Sci. Budapest de R. Eötvös Nom. Sectio Computatorica* (1996).

[Abr 96] Abrial, J.-R.: *B book*. Cambridge University Press, 1996.

[Odi 98] Odifreddi, P.: *Classical Recursion Theory*. Elsevier Science Publ. BV., 1989.

[Koz Var 03] Kozma L. – Varga L.: *A szoftvertechnológia elméleti kérdései*. ELTE Eötvös Kiadó, 2003.