7.1.2. A programozási tétel fogalma

Ha megnézzük az előző fejezet programozási tételeit, azt látjuk hogy ezek feladat-megoldás párokat adnak, a tétel az bennük hogy a megoldásokat a levezetésükkel be is bizonyítjuk. De egy programozási tétel valójában nem egy feladatot és egy megoldást ad meg, hanem a feladat és a megoldás *sémáját*. Ez azt jelenti, hogy a programozási tételekben vannak meg nem határozott részek, amiket a visszavezetéskor lehet „kitölteni". Például az összegzés tételében egy intervallum felett értelmezett függvényről beszélünk, de a függvényről csak a típusát mondjuk meg, egyéb kikötést nem teszünk. Ezek alapján egy programozási tétel a következő elemekből áll:

- programozási feladat (informális)
- paraméterek és a paraméterekre tett kikötések leírása
- formális specifikáció
- megoldó program
- a megoldó program levezetése

7.1. A visszavezetés mint analóg programozási módszer

Analóg programozásnak nevezzük azt, amikor egy feladat megoldásához egy már ismert és megoldott feladat megoldását használjuk fel. Általában nem pontosan ugyanazt a feladatot oldottuk meg korábban, hanem egy hasolót, azonban a két feladat fontosabb részeit meg tudjuk feleltetni egymásnak. Az analóg programozási módszerek feladata a feladatok hasonló részei közötti megfeleltetések alkalmazása a megoldó programokra.

Fontos különbséget tenni a megoldási folyamat megismétlése és a megoldó program lemásolása között. A megoldási folyamat (például levezetés) megismétlése nem igazán hatékony módszer, de előnye hogy kevesebb a hibalehetőség. Ha csak a megoldó programot másoljuk le az új feladatra alkalmazott formában, a két feladat közötti különbségeket nehéz átvezetni a programokra, könnyen tévedhetünk. Ezért gyakran nem konkrét feladatok megoldásait használjuk fel, hanem *sémákat*, vagyis olyan általános feladat-megoldás párokat ahol bizonyos elemek paraméterként vannak megadva, és az egyes konkrét esetek határozzák meg hogy mit kell behelyettesíteni. Ilyen sémákra adunk most néhány példát.

- nevezetes algoritmusok
- nevezetes típusok és adatszerkezetek
- tervminták, elemzési minták
- keretrendszerek (*framework-ök*)

1 Intervallumos függvényes tételek

Néhány egyszerű programozási tétel, mindegyikben az egészek valamely [m…n] intervalluma felett vannak értelmezve. Az üres intervallum jelölése: [n+1…n].

1.1 Összegzés

Tegyük fel, hogy adott az $f\colon [m \ldots n] \to Z$ függvény, és ki akarjuk számolni az $s = \sum_{i=m}^{n} f(i)$ összeget. Először feladat specifikáció:

$$\begin{aligned} A &= \mathbb{Z}_{m} \times \mathbb{Z}_{n} \times \mathbb{Z}_{s} \\ B &= \mathbb{Z}_{m'} \times \mathbb{Z}_{n'} \\ Q &= (m = m' \wedge n = n' \wedge m \leq n+1) \\ R &= (Q \wedge s = \textstyle\sum_{i=m}^{n} f(i)) \end{aligned}$$

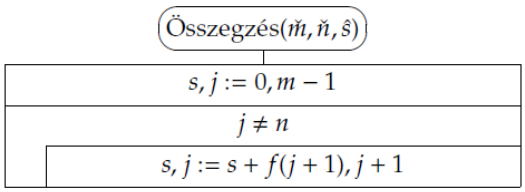
A megoldáshoz az állapotteret bővítjük egy $j\colon Z$ komponenssel. A megoldó program egy ciklus lesz, úgy hogy a ciklus invariánst az utófeltétel gyengítésével kapjuk.

$$\begin{aligned} P &= (Q \wedge s = \textstyle\sum_{i=m}^{j} f(i)) \\ \pi &= (j \neq n) \\ t &= (n - j) \end{aligned}$$

Mivel az invariáns nem következik az előfeltételből, ezért szekvenciát alkalmazunk közbülső feltétellel, ezt nyilván biztosíthatjuk egy $(s, j := 0, m-1)$ értékadással.

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

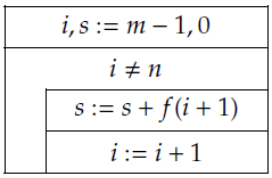
A ciklusmag definiálásához tudjuk, hogy a termináló függvényt csökkenteni kell, így a $(j := j+1)$ értékadást kell elhelyezni a ciklusmagban. Az invariáns tulajdonság megtartása végett az s változóhoz hozzá kell adnunk a $(j+1)$-edik függvényértéket.
A megoldó program a következő:



Megjegyzés:
Egy hasonló programot kaphatunk, ha az összegzést rekurzív függvénynek tekintjük, és az erre vonatkozó programozási tételre vezetjük vissza a feladatot:

$$\begin{aligned} g_{m-1} &= 0, \\ i \geq m-1 : g_{i+1} &= g_i + f(i+1). \end{aligned}$$

Felhasználva a rekurzív függvény kiszámítása programozási tételt, a következő programot kapjuk:



1.2 Számlálás

Ebben az esetben nem egy függvény, hanem egy ß : $[m \ldots n] \to L$ tulajdonság van megadva, és a feladat az, hogy számoljuk meg, hogy az $[m \ldots n]$ intervallum hány elemére teljesül a ß tulajdonság. A feladat specifikációja:

$$A = \underset{m}{\mathbb{Z}} \times \underset{n}{\mathbb{Z}} \times \underset{d}{\mathbb{Z}}$$
$$B = \underset{m'}{\mathbb{Z}} \times \underset{n'}{\mathbb{Z}}$$
$$Q = (m = m' \wedge n = n' \wedge m \leq n + 1)$$
$$R = (Q \wedge d = \textstyle\sum_{i=m}^{n} \chi(\beta(i)))$$

Az állapotteret bővítjük egy komponenssel. Ciklust alkalmazunk és az invariáns az utófeltétel gyengítése:

$$P = (Q \wedge d = \textstyle\sum_{i=m}^{j} \chi(\beta(i)))$$
$$\pi = (j \neq n)$$
$$t = (n - j)$$

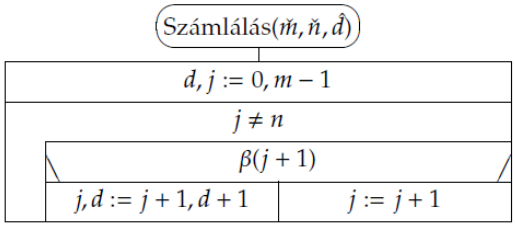
Az invariáns nem teljesül magától, ezért itt is szekvenciát használunk, közbülső feltétellel.

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

A terminálló függvényt itt is csökkenteni kell a ciklusmagban, így szükség van a j növelésére, de ahhoz hogy az invariáns megmaradjon, ebben az esetben a $\chi(\beta(j+1))$-et kellene hozzáadnunk a d-hez, vagyis a következő ciklusmag adódna:

$$j, d := j + 1, \chi(\beta(j + 1))$$

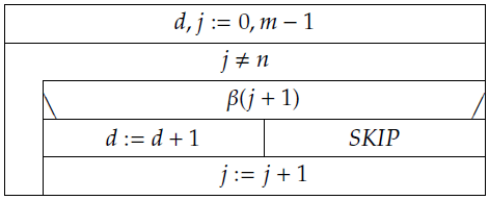
De mivel a $\chi(\beta(j+1))$ kifejezés nem megengedett, ezért elágazást alkalmazunk.
A megoldó program a következő:



Megjegyzés:
Hasonló megoldó programot kaphatunk, ha a számlálást a $\chi \circ \beta$ függvény összegzésére vezetnénk vissza:

- $m \to m$,
- $n \to n$,
- $s \to d$,
- $f(i) \to \chi(\beta(i))$.

Ha erre alkalmazzuk a nem-megengedett kifejezés kitranszformálását, akkor a számláláshoz nagyon hasonló programot kapunk.



1.3 Maximum-keresés

Most tekintsük az $f\colon [m..n] \to H$ függvényt, ahol H tetszőleges rendezett halmaz. A feladat az, hogy határozzuk meg az f egy maximum helyét és a maximum értékét. Vegyük észre, hogy a feladat értelmezhetőségéhez az intervallum nem lehet üres. A feladat specifikációja:

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

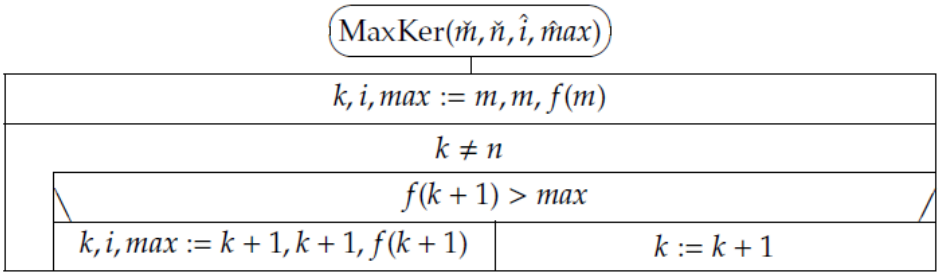
Az állapotteret bővítjük egy k : Z komponenssel. Az invariánst most is az utófeltétel gyengítésével kapjuk, úgy hogy a minden kvantor tartományát szűkítjük:

$$
\begin{aligned}
R &= (Q \wedge i \in [m..k] \wedge max = f(i) \wedge (\forall j \in [m..k] f(j) \leq f(i))) \\
\pi &= (k \neq n) \\
t &= (n - k)
\end{aligned}
$$

Az invariáns kezdeti beállításához most nem lesz jó az üres intervallumos megoldás, mivel az üres intervallumra a maximum nincs értelmezve. Viszont vehetünk egyértelmű intervallumot:

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

ez pedig nyilvánvalóan biztosítható egy (k,i,max := m,m,f(m)) értékadással. Ciklusmagban egyrészt növelni kell a k értékét, hogy csökkentsük a terminálló függvényt. Másrészt nem biztos hogy megmarad az invariáns, ezért elágazást kell használni. A program:



1.4 Feltételes maximum-keresés

Legyen f, mint az előbb, plusz még egy ß tulajdonság is. A feladat most az, hogy [m..n] intervallum ß elemei közül találjuk meg a legnagyobb függvényértékűt. Az intervallum üres is lehet. Nem biztos, hogy van ß tulajdonságú elem. A feladat specifikációja:

$$
\begin{aligned}
A &= \underset{m}{\mathbb{Z}} \times \underset{n}{\mathbb{Z}} \times \underset{i}{\mathbb{Z}} \times \underset{max}{\mathcal{H}} \times \underset{l}{\mathbb{L}} \\
B &= \underset{m'}{\mathbb{Z}} \times \underset{n'}{\mathbb{Z}} \\
Q &= (m = m' \wedge n = n' \wedge m \leq n + 1) \\
R &= (Q \wedge (l = \exists j \in [m..n] : \beta(j)) \wedge \\
&\quad \wedge l \to (i \in [m..n] \wedge max = f(i) \wedge (\forall j \in [m..n] : \beta(j) \to f(j) \leq f(i)))
\end{aligned}
$$

A megoldáshoz az állapotteret bővítjük a k : Z komponenssel. A megoldóprogram egy ciklus lesz, ahol az invariánst a szokott módon kapjuk az utófeltételből, a kvantálás tartományát bővítjük:

$$
\begin{aligned}
P &= (Q \wedge (l = \exists j \in [m..k] : \beta(j)) \wedge \\
&\quad \wedge l \to (i \in [m..k] \wedge max = f(i) \wedge (\forall j \in [m..k] : \beta(j) \to f(j) \leq f(i))) \\
\pi &= (k \neq n) \\
t &= (n - k)
\end{aligned}
$$

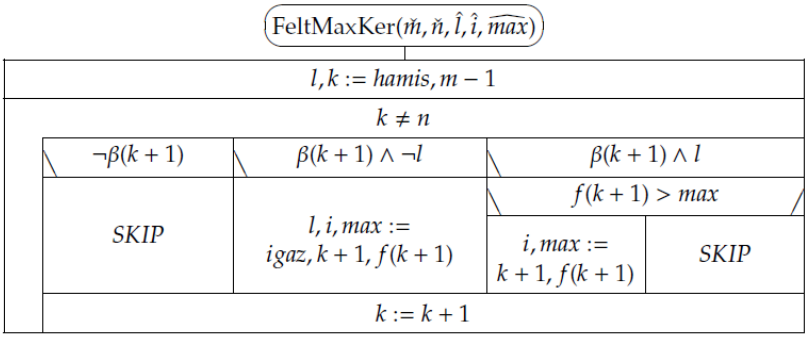
Hogy az invariáns kezdetben teljesüljön, szekvenciát alkalmazunk:

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

A ciklusmagban csökkenteni kell a terminálló függvényt, ezt egy (k:=k+1) értékadással tudjuk megtenni. A ciklusmagban szekvenciát alkalmazunk, a közbülső tagja egy Q'' = $P^{k<-k+1}$, a második tagja az értékadás. A programnak már csak az invariáns tulajdonság fenntartása lesz a feladata, esetszétválasztással a következő esetek adódhatnak:

- nem ß(k+1) az invariáns akkor is megmarad, ha nem csinálunk semmit (SKIP),
- l = hamis, ß(k+1) megtaláltuk az első ß tulajdonságú elemet, (l,i,max:=igaz,k+1,f(k+1)),
- l=igaz, ß(k+1) már találtunk ß tulajdonságú elemet, de ez újabb, ekkor maximum keresést alkalmazunk.

A program:



~~Lineáris keresések~~ Kiválasztás

A lineáris keresések alapfeladata az alábbi függvény segítségével határozható meg:

$$\beta : \mathbb{Z} \to L,\ m \in \mathbb{Z},\ \exists k_0 \geq m : \beta(k_0)$$

A kérdés az, hogy melyik az a legkisebb $i \geq m$, amelyikre teljesül a ß tulajdonság.
Feladatspecifikáció:

$$A = \underset{m}{\mathbb{Z}} \times \underset{i}{\mathbb{Z}}$$
$$B = \underset{m'}{\mathbb{Z}}$$
$$Q = (m = m' \wedge \exists k_0 \geq m : \beta(k_0))$$
$$R = (Q \wedge i \geq m \wedge \beta(i) \forall j \in [m..i-1] : \neg\beta(j))$$

Első és második változat
A megoldáshoz ciklust használunk, a ciklusinvariánst utófeltétel gyengítéssel kapjuk:

$$P = (Q \wedge i \geq m \forall j \in [m..i-1] : \neg\beta(j))$$
$$\pi = (\neg\beta(i))$$
$$t = k_0 - i$$

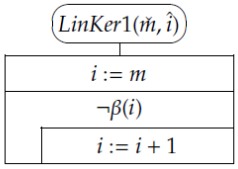
Ha elkezdjük ellenőrizi a ciklus levezetési szabályait, azt találjuk, hogy az első már nem teljesül, nem következik az előfeltételből a $i \geq m$ kikötés. Ezért szekvenciát alkalmazunk.

$$Q' = (Q \wedge i = m)$$

közbülső feltétellel. Az első része (i := m) értékadás lesz. Ciklusmagnak választható az (i := i+1), ellenőrizzük a ciklus levezetési szabályait Q'-re:

$$
\begin{aligned}
Q' &\equiv (Q \wedge i = m) \Rightarrow P \text{ O.K.} \\
P \wedge \neg\pi &\equiv (P \wedge \beta(i)) \Rightarrow R \text{ O.K.} \\
P \wedge \pi &\equiv (P \wedge \neg\beta(i)) \Rightarrow \mathrm{If}((i := i+1), P) \text{ O.K.} \\
P \wedge \pi &\Rightarrow k_0 - i > 0 \text{ O.K.} \\
P \wedge \pi \wedge k_0 - i = t_0 &\Rightarrow \mathrm{If}((i := i+1), k_0 - i < t_0) \equiv \\
&\quad (k_0 - i < t_0)^{i \leftarrow i+1} \equiv (k_0 - i - 1 < t_0) \text{ O.K.}
\end{aligned}
$$

Alábbi program megfelel:



Linker 2.8

Tekintsük az alábbi γ logikai függvényt:

$$\gamma : [m,n] \to \mathbb{L},\ m,n \in \mathbb{Z},\ m \leq n+1$$

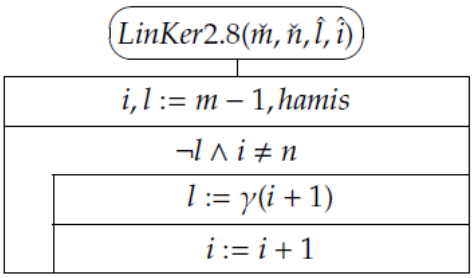
A kérdés most az, hogy az $[m,n]$ intervallumon van-e γ tulajdonságú elem, és ha van, adjuk meg közülük a minimálisat.

Végyük észre, hogy ez visszavezethető[1] az előző esetre, $\delta(i) = (i \leq n)$ választással.

$$
\begin{aligned}
A &= \underset{m}{\mathbb{Z}} \times \underset{n}{\mathbb{Z}} \times \underset{l}{\mathbb{L}} \times \underset{i}{\mathbb{Z}} \\
B &= \underset{m'}{\mathbb{Z}} \times \underset{n'}{\mathbb{Z}} \\
Q &= (m = m' \wedge n = n' \wedge m \leq n+1) \\
R &= (Q \wedge l = (\exists j \in [m,n] : \gamma(j)) \wedge l \Rightarrow (i \in [m,n] \wedge \gamma(i) \wedge \forall j \in [m,i-1] : \neg\gamma(j)))
\end{aligned}
$$

A ciklus levezetése hasonlóan alakul:

$$
\begin{aligned}
P &= (Q \wedge i \in [m-1,n] \wedge l = \exists j \in [m,i] : \gamma(j) \wedge \forall j \in [m,i-1] : \neg\gamma(j)) \\
\pi &= (\neg l \wedge i \neq n) \\
t &= n - i \\
Q' &= (Q \wedge i = m-1 \wedge \neg l)
\end{aligned}
$$



A lineáris keresés egy másik változata a ciklusmagban elágazást alkalmaz, és csak a 'gamma' tulajdonság teljesülése esetén frissíti 'l' értékét.
Egy szintén újabb változat pedig a ciklusmagban csak 'i' értékét növeli az első megfelelő elemig, és 'l' értékét csak a ciklus után állítja be.

6.2.2. Logaritmikus keresés

Legyen $\mathcal{H}$ egy olyan halmaz, amelyen meg van adva egy $\preceq$ teljes rendezési reláció. Tekintsük az $f : [m,n] \to \mathcal{H}$ függvényt. A kérdés az, hogy az f függvény felveszi-e a $h \in \mathcal{H}$ értéket, és ha igen, adjunk meg egy olyan $k \in [m,n$ helyet, ahol a h felvétetik.

'f' monoton növő függvény.

$$
\begin{aligned}
A &= \underset{m}{\mathbb{Z}} \times \underset{n}{\mathbb{Z}} \times \underset{h}{\mathcal{H}} \times \underset{l}{\mathbb{L}} \times \underset{i}{\mathbb{L}} \\
B &= \underset{m'}{\mathbb{Z}} \times \underset{n'}{\mathbb{Z}} \times \underset{h'}{\mathcal{H}} \\
Q &= (m = m' \wedge n = n' \wedge h = h' \wedge m \leq n+1 \wedge \forall j,k \in [m,n] : (j \leq k \Rightarrow f(j) \preceq f(k))) \\
R &= (Q \wedge l = (\exists j \in [m,n] : f(i) = h) \wedge l \Rightarrow (i \in [m,n] \wedge f(i) = h))
\end{aligned}
$$

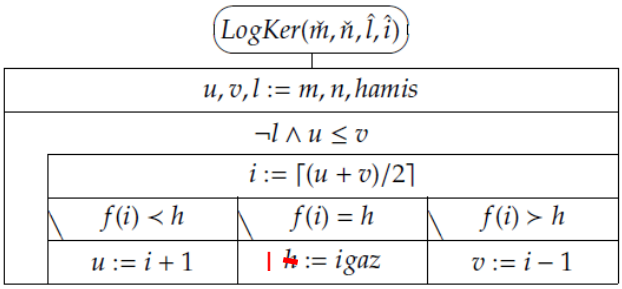
A feladat megoldásához ciklust használunk. Az ötlet most az, hogy nem balról jobbra lépkedünk, hanem alkalmasan szűkítgetjük az intervallumot mindkét oldalról. Az állapotteret bővítjük az $u, v : \mathbb{Z}$ változókkal, ú.h. az $[u,v]$ intervallumra fogjuk szűkíteni a vizsgálatot. Az invariánst az utófeltétel gyengítésével kapjuk, a terminálló függvény most azt fogja megmondani, hogy hány elemet kell még megvizsgálni. Hogy a ciklus kezdeti feltételei teljesüljenek, az inicializációt egy szekvenciával végezzük el.

$$
\begin{aligned}
P &= (Q \wedge u \geq m \wedge v \leq n \wedge \forall j \in [m,n] \setminus [u,v] : f(j) \neq h \wedge \\
&\quad \wedge\, l \Rightarrow (i \in [u,v] \wedge f(i) = h)) \\
\pi &= (\neg l \wedge u \leq v) \\
t &= (v - u + 1) \\
Q' &= (Q \wedge u = m \wedge v = n \wedge l = \mathit{hamis})
\end{aligned}
$$

A ciklusmagban nincs más dolgunk, mit „kettévágni" az éppen vizsgált $[u,v]$ intervallumot. Elvileg bárhol elvégezhetnénk ezt a vágást, de az eljárás *műveletigénye* szempontjából a legelőnyösebb, ha pont a közepén. A ciklusmag egy szekvencia, melynek közbülső feltétele:

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

A specifikáció ilyetén finomítása után könnyen ellenőrizhető, hogy az alábbi program megoldja a feladatot:



5.1.3. Rekurzív függvény

Legyen most f az alábbi rekurzív összefüggésekkel megadva:

$$\begin{aligned} f_m &= t_0 \\ f_{m-1} &= t_{-1} \\ &\vdots \\ f_{m-k+1} &= t_{-k+1} \end{aligned}$$

Itt a k rögzített egész szám, $t_0, t_{-1}, \ldots, t_{-k+1}$ konstansok. A rekurzív összefüggést az alábbi képlettel adjuk meg:

$$i \geq m : f_{i+1} = F(i+1, f_i, f_{i-1}, \ldots, f_{i-k+1})$$

Ezekben a képletekben használtuk az $f_i = f(i)$ jelölést. A feladatot az állapottér bővítésével oldjuk meg:

$$A' = \underset{x}{X} \times \underset{y}{Y} \times \underset{y_1}{Y} \ldots \underset{y_{k-1}}{Y} \times \underset{i}{\mathbb{Z}}$$

A megoldó program egy ciklus lesz:

$$\begin{aligned} P &= (i \in [m,n] \wedge y = f_i \wedge y_1 = f_{i-1} \wedge \ldots \wedge y_{k-1} = f_{i-k+1}) \\ \pi &= (i \neq n) \\ t &= (n - i) \end{aligned}$$

Mivel az előfeltételből nem következik a ciklusinvariáns, szekvenciát alkalmazunk a következő közbülső feltétellel:

$$Q' = (Q \wedge i = m \wedge y = t_0 \wedge y_1 = t_{-1} \wedge \ldots \wedge y_{k-q} = t_{-k+1})$$

$i, y, y_1, \ldots, y_{k-1} = m, t_0, t_{-1}, \ldots, t_{-k+1}$
$i \neq n$
$y, y_1, \ldots, y_{k-1} := F(i+1, y, y_1, \ldots, y_{k-1}),$ $y, y_1, \ldots, y_{k-2}$
$i := i + 1$

A ciklusmag közbülső feltételének természetesen használhatjuk a $P^{i \leftarrow i+1}$ feltételt. Általában $k = 1$ szokott lenni, ekkor a program az alábbi alakra egyszerűsödik:

$i, y := m, t_0$
$i \neq n$
$y := F(i+1, y)$
$i := i + 1$

4.2. Program-transzformációk

A programtranszformációk általában olyan átalakításai egy programnak (nem feltétlenül azonos állapottér felett), hogy a transzformált program az állapottér egy lényeges része felett ekvivalens lesz az eredeti programmal. Ennél kevesebbel is megelégedhetünk, bizonyos transzformációknál olyan programot kapunk, ami az eredeti program által megoldott bármely feladatot megoldja, vagyis az eredeti programnak finomítása. (ez GT-nél is hasonlóan szerepel.)

Emlékezzünk arra a tételre, amit az első fejezetben mondtunk ki a programfüggvényt mint feladatot megoldó programról:

$$\hat{S} \text{ mo. } p(S) \Leftrightarrow \left(\forall F, S \text{ mo. } F \Rightarrow \hat{S} \text{ mo. } F\right)$$

Ezt nevezhetjük akár a programtranszformációk alaptételének is. Általában ez felhasználható a programtranszformációk helyességének bizonyítására. Azonban, ha egy transzformációról „látszik, hogy jó", a bizonyítástól eltekintünk.

4.2.1. Szerepe a feladat finomításában

Csak a levezetési szabályok használatával megoldhatunk egy programot, de ha közben arra is oda kell figyelni, hogy mindenhol megengedett kifejezéseket és feltételeket használjunk, akkor a munka kínkeserves lesz. A feladat finomításának hasznos eszközei az adat- és függvényabsztrakció. A függvényabsztrakciónál nemmegengedett kifejezéseket írunk fel, miközben programkonstrukciókkal bontjuk szét a feladatot. Ezeket tudjuk megengedetté tenni a *kiemelési* szabályok segítségével. Az adatabsztrakciónál, ha a megoldóprogram előállítása során az állapottérbe sorozatokat vezettünk be, gyakran kapunk esetlen, a sok átmeneti tároló miatt nem-hatékony programokat. Ezeknek a programokat az adatfeldolgozási tulajdonságaiból kiindulva a *programinverzió* segítségével az átmeneti változókat eliminálhatjuk az állapottérből, ezzel egy tömörebb és hatékonyabb (bár lehet, hogy kevésbé átlátható) programot kapunk. A programinverzió különösen akkor kap jelentőséget, ha egy absztrakt tipust elő- vagy utófeldolgozással implementáltunk.