Programozási tételek összeépítése

Szlávi Péter

2009-2012

Bevezetés

Három kulcsfogalom

- Programozási tétel [1/1-6. o.]
- Programtranszformáció [2/1-10. o.]
- Ciklusinvariáns [3/19. o.]

Az első két fogalom „logikája” nagyjából azonos: egy állítás, amely két dolog „egybecsengéséről szól”. A *programozási tétel* egy feladatspecifikációt és egy absztrakt algoritmust kapcsol egybe, kimondván, hogy „a specifikációban megfogalmazott feladatot az algoritmus megoldja”, addig a *programtranszformáció* két algoritmus ekvivalenciáját állítja bizonyos feltételek teljesülése esetén.

A *ciklusinvariáns* olyan állítás, amely a ciklus tetszőleges számú lefutása során állandóan igaz marad. Ez az állandóság természetesen a ciklusban változó adatok ellenére igaz. Éppen ez az a tulajdonsága, amely képessé teszi arra, hogy a ciklus által elvégzett transzformációt leírja. Szokás a ciklusinvariánst a ciklus nyitó kulcs-szava és a feltételvizsgálatot bevezető ’amíg’ közé képzelni:

```
Ciklus [ciklusinvariáns] amíg ciklus feltétel …
```

Mivel az elől-feltételes ciklussal bármelyik ciklus kifejezhető, így a fenti alapján bármilyen ciklust tartalmazó programdarabra értelemszerűen alkalmazható a fenti „elképzelés”.

Jelölések

A tételekre az adott feladat megoldásának felvázolásakor *függvény*ként hivatkozunk ([1]-beli szellemben).

A „levezetés módszere”

A feladatok megoldásánál hozzávetőlegesen az alábbi sablont követjük:

1. elkészítjük a feladat specifikációját;
2. elemezzük ezt, hogy felfedezzük benne a megoldáshoz felhasználandó programozási tételeket;
3. ez alapján elkészítjük a megoldás 0. változatának a vázlatát (tételfüggvényekkel);
4. kifejtjük a hivatkozott tételek algoritmusait (azok paramétereinek aktualizálása után), hogy megkaphassuk a 0. megoldást;

   A kapott algoritmus garantáltan helyes megoldása a feladatnak, csak éppen nem biztos, hogy a legegyszerűbb.
5. észrevételt teszünk az „ésszerűsítésre” vonatkozólag;
6. végrehajtjuk az észrevételnek megfelelő program-átalakításokat, miközben ügyelünk a helyesség megmaradásra.

Feladatok

Megadjuk előzetesen a feladatok szövegét:

a) Adott egy számsorozat. Határozzuk meg T-tulajdonságú elemeinek átlagát!

b) Adott egy sorozat. Határozzuk meg T-tulajdonságú elemeinek maximumát!

c) Adjuk meg egy számsorozat minimális, maximális elemét, valamint átlagát!

d) Egy sorozatban hány legnagyobb elem van?

e) Van-e a sorozatban legalább K darab T-tulajdonságú elem?

f) A sorozat legnagyobbika nagyobb-e, mint egy előre adott érték?

Minden olyan feladatban, amelyben felbukkan egy T-*tulajdonság*, feltesszük, hogy a megvalósításaként olyan függvényt használunk, amely *x-invariáns*. [2/4. o.]

Az a) feladat megoldása

Adott egy számsorozat. Határozzuk meg T-tulajdonságú elemeinek átlagát!

a.1. Specifikáció

Be: $N \in \mathbb{N}$, $X \in \mathbb{R}^*$ $[T:\mathbb{R}\rightarrow L]$

Ki: $Átl \in \mathbb{R}$

Ef: $N=Hossz(X) \wedge N>0 \wedge \exists i \in [1..N]: T(X_i)$

Uf: $dbT=\sum_{i=1}^{N} \chi(T(X_i)) \wedge Átl=\sum_{\substack{i=1 \\ T(X_i)}}^{N} X_i \,/dbT$ ♣

a.2. A tételek felismerése

„Ránézésre” a *megszámolás* és a *sorozatszámítás* (összegzés) *tétel* darabjai ismerhetők fel. A második szummás kifejezés „szokásos alakúvá” transzformálásához az Uf alábbi átalakítása szükséges:

$\sum_{\substack{i=1 \\ T(X_i)}}^{N} X_i = \sum_{i=1}^{dbT} xT_i$, ahol bevezettük a dbT méretű xT nevű segédtömböt az alábbi jelentéssel:

$xT \in \mathbb{R}^{dbT} \wedge xT \subseteq X \wedge \forall i \in [1..dbT]: T(xT_i)$

Így már a megszámolás helyett a *kiválogatás* tételt juttatja eszünkbe. A módosított Uf „egyben”:

$dbT=\sum_{i=1}^{N} \chi(T(X_i)) \wedge xT \in \mathbb{R}^{dbT} \wedge xT \subseteq X \wedge \forall i \in [1..dbT]: T(xT_i)$ $\wedge$ $Átl=\sum_{i=1}^{dbT} xT_i \,/dbT$

kiválogatás *összegzés*

a.3. A 0. megoldás vázlata

```
(dbT,xT):=Kiválogatás(N,X,T)
Átl:=Sorozatszámítás(dbT,xT,0,+)/dbT
```

a.4. A 0. megoldás kifejtése

Bevezetünk egy `s` segédváltozót ügyelve arra, hogy neve nehogy egyezzen valamely már használt változóéval, ui. ezzel esetleg nem várt „interferenciát” okoznánk. (Nyilvánvalóan a ciklusváltozók választását is az előbbi szempont korlátozza.)

♣ Itt mind a két féle „T-vel szűkített szummá”-s kifejezés szerepel. Nyilvánvaló a kapcsolat közöttük:

$\sum_{\substack{i=1 \\ T(X_i)}}^{N} X_i = \sum_{i=1}^{N} \chi(T(X_i)) * X_i$, illetve $\sum_{\substack{i=1 \\ T(X_i)}}^{N} 1 = \sum_{i=1}^{N} \chi(T(X_i))$

α
```
dbT:=0
Ciklus i=1-től N-ig
  Ha T(X(i)) akkor dbT:+1; xT(dbT):=X(i)
Ciklus vége
```

β
```
s:=0
Ciklus i=1-től dbT-ig
  s:+xT(i)
Ciklus vége
Átl:=s/dbT
```

a.5. Észrevétel

Kigyűjtéskor rögvest összegezhetnénk is.

a.6. Programátalakítások

A két ciklus „mechanikus” egyesítése nem megy. Ezért 1) meg kell oldani az azonos ciklusszervezést (végérték-hangolást); 2) be kell látni a ciklusmagok függetlenségét; 3) a ciklusokat egyesítjük.

1) Esélyes a β-ciklusban az „N-ig”-re áttérés. De szemantikus azonosság érdekében az α-ciklusmag-beli feltételt át kell venni, és az xT-ben való haladást itt is nyomon kell követni, s a pontos átalakíthatóság kedvéért „szemet bántó” (elágazás-) szétbontásra is kényszerülünk az α-ciklusmagban (PT16 „visszafelé”: [2/10. o.]).

2) A két ciklus formális függetlenségének fenntartása érdekében új segédváltozót (db2) vezetünk be a β ciklusmagjában. Teljes függetlenség nem áll fönn, hiszen a β ciklusban az α ciklusban képződő xT-ket felhasználjuk. A függés azonban „laza”: ui. a szinkronba hozott β ciklus adott ciklislépésben éppen azt az xT-t használja föl, amelyiket az α ciklus éppen előállított:

α
```
dbT:=0
Ciklus i=1-től N-ig
  Ha T(X(i)) akkor dbT:+1
  Ha T(X(i)) akkor xT(dbT):=X(i)
Ciklus vége
```

β
```
s:=0; db2:=0
Ciklus i=1-től N-ig
  Ha T(X(i)) akkor db2:+1
  Ha T(X(i)) akkor s:+xT(db2)
Ciklus vége
Átl:=s/dbT
```

3) Az egyesítésnek nincs akadálya, mivel az α-ciklusmag ugyanabban a ciklislépésben „elkészíti” a β-ciklusmagban felhasznált xT-elemet. Így a PT10 [2/8. o.] programtranszformációval kapjuk:

α&β
```
dbT:=0; s:=0; db2:=0
Ciklus i=1-től N-ig
  Ha T(X(i)) akkor dbT:+1
  Ha T(X(i)) akkor xT(dbT):=X(i)
  Ha T(X(i)) akkor db2:+1
  Ha T(X(i)) akkor s:+xT(db2)
Ciklus vége
Átl:=s/dbT
```

A kékkel jelölt részek azonos működése (továbbá T x-invarianciája) lehetővé tesz nyilvánvaló egyszerűsítéseket: a dbT és db2 közül a dbT tartjuk meg, s db2 helyett is őt használjuk:

```
      dbT:=0; s:=0
      Ciklus i=1-től N-ig
        Ha T(X(i)) akkor dbT:+1
α&β     Ha T(X(i)) akkor xT(dbT):=X(i)
        Ha T(X(i)) akkor s:+xT(dbT)
      Ciklus vége
      Átl:=s/dbT
```

A PT16 [2/10. o.] transzformáció feltételei fennállnak, így alkalmazhatjuk („duplán”) az elágazások egyesítésére:

```
      dbT:=0; s:=0
      Ciklus i=1-től N-ig
α&β     Ha T(X(i)) akkor dbT:+1; xT(dbT):=X(i); s:+xT(dbT)
      Ciklus vége
      Átl:=s/dbT
```

Józanésszel is nyilvánvaló, hogy az xT tömb(elem) közvetítése nélkül is megy, de éppen ezt formalizálja a PT1 [2/6. o.] transzformáció:

```
      dbT:=0; s:=0
      Ciklus i=1-től N-ig
α&β     Ha T(X(i)) akkor dbT:+1; s:+X(i)
      Ciklus vége
      Átl:=s/dbT
```

Így kaptuk meg azt, amit józanésszel is kaptunk volna. Csak éppen most minden lépést bizonyítottan helyes transzformációval vezérelve tettünk meg, így tökéletesen nyugodtak lehetünk a programhelyessége miatt.

A b) feladat megoldása

Adott egy sorozat. Határozzuk meg T-tulajdonságú elemeinek maximumát!

b.1. Specifikáció

Be: $N\in\mathbb{N},\ X\in H^*\ [T:H\to L,\ \le:H\cup\{-\infty\}\times H\cup\{-\infty\}\to L]$

Ki: $Max\in H\cup\{-\infty\}$

Ef: $N=Hossz(X)\ \wedge\ RendezettHalmaz_{\le}(H\cup\{-\infty\})\ \wedge\ \forall h\in H: -\infty<h$

Uf: $$dbT=\sum_{i=1}^{N}\chi(T(X_i))\ \wedge$$
$$((dbT=0)\to Max=-\infty)\ \wedge$$
$$((dbT>0)\to(Max\in X\wedge T(Max)\wedge\forall i\in[1..N]:(T(X_i)\to X_i\le Max)))$$

b.2. A tételek felismerése

Az Uf-ben „ránézésre” a *megszámolás* és a *maximum(érték-)kiválasztás tétel* darabjai ismerhetők fel. Az előbbi feladat megoldása után a megszámolás helyett *kiválogatás*ra helyesbítünk. Az Uf alábbi átalakításával kicsit nagyobb „összhang” teremthető az említettekkel:

$$\boxed{dbT=\sum_{i=1}^{N}\chi(T(X_i))\wedge xT\in\mathbb{R}^{dbT}\wedge xT\subseteq X\ \wedge\ \forall i\in[1..dbT]:T(xT_i)}\ \wedge\ ((dbT=0)\to Max=-\infty)\ \wedge\ ((dbT>0)\to\boxed{(Max\in xT\ \wedge\ \forall i\in[1..dbT]:xT_i\le Max)})$$

kiválogatás — *maximum(érték-)kiválasztás*

b.3. A 0. megoldás vázlata

```
(dbT,xT):=Kiválogatás(N,X,T)
Ha dbT=0 akkor
  Max:=-∞
különben
  Max:=MaximumÉrtékKiválasztás(dbT,xT,≤)
Elágazás vége
```

b.4. A 0. megoldás kifejtése

```
α       dbT:=0
        Ciklus i=1-től N-ig
          Ha T(X(i)) akkor dbT:+1; xT(dbT):=X(i)
        Ciklus vége
        Ha dbT=0 akkor
          Max:=-∞
        különben
          Max:=xT(1)
β         Ciklus i=2-től dbT-ig
            Ha Max<xT(i) akkor Max:=xT(i)
          Ciklus vége
        Elágazás vége
```

b.5. Észrevétel

Kigyűjtéskor rögvest vizsgálhatnánk a „nagyobbságot” is.

b.6. Programátalakítások

A két ciklus „mechanikus” egyesítése nem megy. Ezért 1) meg kell oldani az azonos ciklusszervezést (kezdő- és végérték-hangolást); fel kell oldani az előfordulás „hierarchiaszint-beli” különbözőségének a problémáját; 2) be kell látni a ciklusmagok függetlenségét; 3) majd az egyesítés.

1) Kezdjük a hierarchiaszint-különbözőség feloldásával, az elágazás megszüntetésével! Első lépésben az elágazás két ágát „közelítjük” egymáshoz:

```
        Max:=-∞ Max:=xT(1)
β       Ciklus i=1 2-től dbT-ig
        …
```

Hogy a β ezen átírása szemantikusan ekvivalens programot eredményez, az Ef-beli azon elvárás miatt nyilvánvalóan teljesül: ∀h∈H: -∞<h.

Folytassuk az átalakítást!

```
        Ha dbT=0 akkor
          Max:=-∞
        különben
          Max:=-∞
β         Ciklus i=1-től dbT-ig
            Ha Max<xT(i) akkor Max:=xT(i)
          Ciklus vége
        Elágazás vége
```

Vegyük észre, hogy a PT17 [2/10. o.] alkalmazható, így kapjuk:

```
        Max:=-∞
        Ha dbT=0 akkor
          Max:=-∞
        különben
          Max:=-∞
β         Ciklus i=1-től dbT-ig
            Ha Max<xT(i) akkor Max:=xT(i)
          Ciklus vége
        Elágazás vége
```

Ami nyilvánvalóan ugyanaz, mint

```
        Max:=-∞
        Ha dbT>0 akkor
          Ciklus i=1-től dbT-ig
β           Ha Max<xT(i) akkor Max:=xT(i)
          Ciklus vége
        Elágazás vége
```

Mivel azonban dbT=0 esetben a ciklus üresen fut le, az elágazás megszüntethető. Így az egy szintre hozást elértük:

```
        Max:=-∞
        Ciklus i=1-től dbT-ig
β         Ha Max<xT(i) akkor Max:=xT(i)
        Ciklus vége
```

A Max:=-∞ PT2 [2/6. o.] miatt a legelőre vihető:

```
        Max:=-∞
        dbT:=0
α       Ciklus i=1-től N-ig
        …
        Ciklus i=1-től dbT-ig
β       …
```

A két ciklus ciklusváltozójának a kezdőértéke már azonossá lett. A végértékre kell fokuszálnunk. Esélyes a β-ciklusban az „N-ig”-re áttérés. De a szemantikus azonosság érdekében az α-ciklusmag-beli feltételt át kell venni, és az xT-ben való haladást itt is nyomon kell követni, s a pontos átalakíthatóság kedvéért „szemetbántó” szétbontásra is kényszerülünk az α-ciklusmagban (PT16 „visszafelé”: [2/10. o.]).

2) A két ciklus formális függetlenségének fenntartása érdekében új segédváltozót (db2) vezetünk be β ciklusmagjában.

```
        Max:=-∞
        dbT:=0
        Ciklus i=1-től N-ig
α         Ha T(X(i)) akkor dbT:+1
          Ha T(X(i)) akkor xT(dbT):=X(i)
        Ciklus vége
        db2:=0
        Ciklus i=1-től N-ig
β         Ha T(X(i)) akkor db2:+1
          Ha T(X(i)) akkor Ha Max<xT(db2) akkor Max:=xT(db2)
        Ciklus vége
```

3) Az egyesítésnek nincs akadálya, mivel az α-ciklusmag ugyanabban a ciklusléptésben „elkészíti” a β-ciklusmagban felhasznált xT-elemet. Így a PT10 [2/8. o.] programtranszformációval kapjuk (a kvázi közös utasításokat persze csak egyetlen példányban visszük tovább, követve az előző feladat hasonló technikáját):

```
        Max:=-∞
        dbT:=0
        Ciklus i=1-től N-ig
α&β       Ha T(X(i)) akkor dbT:+1
          Ha T(X(i)) akkor xT(dbT):=X(i)
          Ha T(X(i)) akkor Ha Max<xT(dbT) akkor Max:=xT(dbT)
        Ciklus vége
```

A PT16 [2/10. o.] transzformáció feltételei fennállnak, így alkalmazhatjuk („duplán”) az elágazások összevonására:

```
        Max:=-∞
-----------------------------------------------------
        dbT:=0
        Ciklus i=1-től N-ig
          Ha T(X(i)) akkor dbT:+1; xT(dbT):=X(i)
α&β                        Ha Max<xT(dbT) akkor Max:=xT(dbT)
          Elágazás vége
        Ciklus vége
```

Józanésszel is nyilvánvaló, hogy az xT tömb(elem) nélkül is megy, de éppen ezt formalizálja a PT1 [2/6. o.] transzformáció:

```
        Max:=-∞
-----------------------------------------------------
        dbT:=0
        Ciklus i=1-től N-ig
          Ha T(X(i)) akkor dbT:+1
α&β                        Ha Max<X(i) akkor Max:=X(i)
          Elágazás vége
        Ciklus vége
```

Sőt a dbT-re sincs már szükség (a számításhoz sem!), ezért elhagyhatjuk:

```
        Max:=-∞
-----------------------------------------------------
        Ciklus i=1-től N-ig
α&β       Ha T(X(i)) akkor Ha Max<X(i) akkor Max:=X(i)
        Ciklus vége
```

Nem szimmetrikus logikai műveletek [2/5. o.] esetén az alábbi megoldás is elképzelhető, amelyet a PT7 formalizálja [2/7. o.]:

```
        Max:=-∞
-----------------------------------------------------
        Ciklus i=1-től N-ig
α&β       Ha T(X(i)) és Max<X(i) akkor Max:=X(i)
        Ciklus vége
```

A c) feladat megoldása

Adjuk meg egy számsorozat minimális, maximális elemét, valamint átlagát!

Gyakorlásképpen oldjuk meg a fenti sablont követve, és az előző két példa megfontolásait ismerve.

A d) feladat megoldása

Egy sorozatban hány legnagyobb elem van?

d.1. Specifikáció

Be: $N \in \mathbb{N}, X \in H^* \quad [T:H \to L, \leq: H \times H \to L]$

Ki: $db \in \mathbb{N}$

Ef: $N = Hossz(X) \wedge N \geq 1$

Uf: $db = \sum_{i=1}^{N} \chi(X_i = max) \quad \wedge \quad max \in X \wedge \forall i \in [1..N]: X_i \leq max$

d.2. A tételek felismerése

Az Uf-ben a *megszámolás* és a *maximum(érték)kiválasztás tétel* darabjai ismerhetők fel. Az Uf alábbi bontásával nyilvánvalóvá tehető érzésünk:

$$db=\sum_{i=1}^{N}\chi(X_i = max) \quad \wedge \quad max\in X \wedge \forall i\in[1..N]: X_i\leq max$$

megszámolás *maximum(érték)kiválasztás*

d.3. A 0. megoldás vázlata

Az világos, hogy a megszámolásban szereplő `max` előbb számítandó ki:

```
max:=MaximumÉrtékKiválasztás(N,X,≤)
db:=Megszámolás(N,X,=max)
```

d.4. A 0. megoldás kifejtése

α
```
max:=X(1)
Ciklus i=2-től N-ig
  Ha max<X(i) akkor max:=X(i)
Ciklus vége
```

β
```
db:=0
Ciklus i=1-től N-ig
  Ha X(i)=max akkor db:+1
Ciklus vége
```

d.5. Észrevétel

A két „hasonszőrű” ciklus esetleges egyesítése rövidíthetné a megoldást.

d.6. Programátalakítások

A 1) ciklusok azonos szervezésűvé alakítása után 2) be kell látnunk, hogy a ciklusmagok függetlenek, s ez esetben a 3) egyesítés helyes eredményre vezet.

1) Két út kínálkozik: a) a β-beli ciklus 2-től, vagy b) az α-beli 1-től.

a) az i=1-re a ciklusmag még a ciklus előtt végrehajtandó:

β
```
Ha X(1)=max akkor  db:=1
           különben db:=0
Ciklus i=2-től N-ig
  Ha X(i)=max akkor db:+1
Ciklus vége
```

b) igaz, hogy hatékonyság szempontjából némileg „pazarló”, de szemantikusan (az Ef miatt) ekvivalens:

α
```
max:=X(1)  [esetleg: -∞;  ez esetben az Ef második tagja elhagyható!]
Ciklus i=1-től N-ig
  Ha max<X(i) akkor max:=X(i)
Ciklus vége
```

2) A két ciklusmag távolról sem független! Hiszen amint az első ciklusmagban az elágazás igazzá válik, megváltozik a második ciklusmagban szereplő feltétel, az eddigi számlálásunk érvénytelenné válik. Hogy érvényessé tegyük, tabularázát csinálunk: a db-t inicializáljuk. Így paradox módon a függőség még szorosabbá tételével érünk el eredményt.

Az előbbi b) esetet követve, kapjuk:

α&β
```
max:=X(1); db:=0
Ciklus i=1-től [*] N-ig
  Ha max<X(i) akkor max:=X(i); db:=0
  Ha X(i)=max akkor db:+1
Ciklus vége
```

A `[*]` jelölésre később lesz szükség.

3) Vizsgáljuk meg a kapott algoritmus helyességét!

Be kell látni, hogy a ciklusmag lefutásakor az aktuális `i`-t megelőző elemig `db` számú maximális érték volt-e az `X`-ben, azaz a **ciklusinvariáns**t (`[*]`-nál elképzelve):

$$db=\sum_{j=1}^{i-1}\chi(X_j=max) \quad \wedge \quad max\in X \wedge \forall j\in[1..i): X_j\le max$$

i=1-re nyilvánvaló. Feltehető indukciós feltételül, hogy általános `i`-ig teljesül. Azt kell belátni, hogy a következő ciklusléptésben az állítás változatlanul igaz marad.

Három esetet kell megkülönböztetni: (i) `max<X(i)`, (ii) `max=X(i)`, ill. (iii) `max>X(i)`.

(i) esetben az első feltétel (s a transzformációja által a második is!) igaznak adódik, ennek következtében a `max` felveszi az új maximális értéket, és a `db` 1 lesz;

(ii) esetben az első elágazás „üres", a másodikban elvárásainknak megfelelően eggyel nő a `db`;

(iii) esetben mindkét elágazás „üresen tevékenykedik", minden marad a „régiben", ahogy ez esetben el is várjuk.

Megjegyzés: az átalakítás „értelme" a hatékonyság szempontjából akkor lesz igen kézzelfogható, amikor a sorozat elemeihez való hozzáférés korlátozott, esetleg nagy időigényű. Ez az eset pl. szekvenciális fájlok esetén. Ebben az esetben a fájl kétszeres végig olvasása jelentős többletidőt visz el az utóbbi megoldáshoz képest. Igaz, ilyenkor az `X(i)`-re hivatkozás csak egyszer „jogos", a beolvasáskor: egy lokális adatba. A többi helyen e lokális adatra történika hivatkozás.

Az e) feladat megoldása – intuitíve, helyességbizonyítással

Van-e a sorozatban legalább K darab T-tulajdonságú elem?
Ebben a megközelítésben megmutatjuk, hogy van olyan eset, amikor nem programozási tételek alapján, hanem „érzésre" alkotunk meg egy hatékony megoldást. Mivel ekkor nem bizonyítottan helyes átalakításokon keresztül jutunk el egy (feltételezett) megoldáshoz, nem lehetünk meggyőződve a helyességről. Emiatt kényszerülünk matematikai bizonyításra. Most erre látunk példát.

e.1. Specifikáció

Be: $N\in\mathbb{N}, X\in H^*, K\in\mathbb{N}$ $[T:H\to L]$

Ki: $VanK\in L$

Ef: $N=Hossz(X) \wedge N\ge K$ [egyébként triviális a megoldás]

Uf: $VanK=\sum_{i=1}^{N}\chi(T(X_i))\ge K$

e.2. A tételek felismerése

Az Uf-ben csak a megszámolás tétel ismerhető fel:

Uf: $VanK=db\ge K \wedge db=\sum_{i=1}^{N}\chi(T(X_i))$

e.3. A 0. megoldás vázlata

```
db:=Megszámolás(N,X,T)
VanK:=db≥K
```

e.4. A 0. megoldás kifejtése

α

```
db:=0
Ciklus i=1-től [*] N-ig
  Ha T(X(i)) akkor db:+1
Ciklus vége
VanK:=db≥K
```

A `[*]` jelölésre később lesz szükség.

e.5. Észrevétel

Zavaró, hogy az N. elemig el kell menni a ciklusban, pedig lehet, hogy már sokkal hamarabb kiderült a válasz.

e.6. Programátalakítások

Megoldás: az „előbb-kilépés"-t lehetővé tevő feltétel beiktatása a ciklusfeltételbe. Ehhez 1) először ekvivalens amíg-os ciklussá kell alakítani a megszámolás ciklusát, majd a 2) módosított feltételű ciklussal megfogalmazott megoldás 3) ekvivalenciát kell belátni.

1) amíg-os ciklussá átalakítás:

```
Ciklus i=1-től [*] N-ig          ↔      i:=1
  …                                     Ciklus [*] amíg i≤N
Ciklus vége                               …
                                          i:+1
                                        Ciklus vége
```

Az ekvivalencia belátása most nem probléma, hiszen az utóbbival definiáltuk az előbbi szemantikáját.

2) a ciklusfeltétel bővítése után kapott változat:

β

```
db:=0; i:=1
Ciklus [*] amíg i≤N és db<K
  Ha T(X(i)) akkor db:+1
  i:+1
Ciklus vége
VanK:=db≥K
```

3) az α és β megoldások ekvivalenciájának a belátásához csak a ciklusinvariánsokra van szükség. A ciklusinvariáns a `[*]` ciklusfeltétel-vizsgálatánál azonosan igaz állítás:

$$\text{Ciklusinvariáns}_\alpha = \left(db = \sum_{j=1}^{i-1} \chi(T(X_j)) \right) = \text{Ciklusinvariáns}_\beta$$

Ezt –ujjgyakorlatként– lássa be mindkét algoritmusra!

A ciklusból kilépéskor igaz állítás: *ciklusinvariáns* ∧ ¬*ciklusfeltétel* [4/30. diától].

Ciklusból való kilépés után:

α

$$\left(db = \sum_{j=1}^{i-1} \chi(T(X_j)) \right) \wedge \neg(i \le N) \equiv$$

$$\left(db = \sum_{j=1}^{i-1} \chi(T(X_j)) \right) \wedge (i = N+1) \equiv$$

$$\left(db = \sum_{j=1}^{N} \chi(T(X_j)) \right)$$

β

$$\left(db = \sum_{j=1}^{i-1} \chi(T(X_j)) \right) \wedge \neg(i \le N \wedge db < K) \equiv$$

$$\left(db = \sum_{j=1}^{i-1} \chi(T(X_j)) \right) \wedge (i = N+1 \vee db = K) \equiv$$

$$\left(db = \sum_{j=1}^{i-1} \chi(T(X_j)) \wedge i = N+1 \right) \vee \left(db = \sum_{j=1}^{i-1} \chi(T(X_j)) \wedge db = K \right) \equiv$$

$$\left(db = \sum_{j=1}^{N} \chi(T(X_j)) \right) \vee \left(\sum_{j=1}^{i-1} \chi(T(X_j)) = K \right)$$

Ezután vizsgáljuk meg az eredmény (a `VanK` értéke) szempontjából érdekes két esetet: (i) nincs `K` darab `T`-tulajdonságú, ill. (ii) legalább `K` darab `T`-tulajdonságú van.

(i) Az α-esetben `db<K` fog teljesülni.

A β-esetben a feltétel első része igaz, azaz itt is a `db<K` fog teljesülni.

Így tehát mindkét esetben a `VanK` azonos módon hamisra fog állítódni.

(ii) Az α-esetben `db≥K` fog teljesülni.

A β-esetben a feltétel második része igaz (az első lehet, hogy hamis), azaz `db=K` fog teljesülni.

Így tehát mindkét esetben a `VanK` igazra fog állítódni.

Az e) feladat megoldása – tisztán, programtranszformációkkal

Van-e a sorozatban legalább K darab T-tulajdonságú elem?
Az előbb megoldott feladatot most szigorúan a programtranszformációkra építjük.

e.1. Specifikáció

Az utófeltételnek olyan alakját is megfogalmazhatjuk, amelyben már megbújik az a szándék, hogy a T-tulajdonságú elemek számlálását a lehető legkorábban abbahagyjuk.

Uf: $\mathrm{VanK}=\exists i\in[1..N]: \sum_{j=1}^{i}\chi(T(Xj)) = K$

Hogy világosabban felismerhető legyenek a benne rejlő programozási tételek, szétbontjuk a kvantorizált logikai kifejezést, amihez bevezetünk egy új függvényt:

Uf: $\mathrm{VanK}=\exists i\in[1..N]:\mathrm{HányT}(X,i)=K$

Def: $\mathrm{HányT}: H^{*}\times\mathbb{N}\rightarrow\mathbb{N}$

$\mathrm{HányT}(x,i):= \sum_{j=1}^{i}\chi(T(Xj))$

E felírásból nyilvánvaló, hogy egy eldöntés tétel, valamint egy bele épülő megszámolás tétel alkalmazásával jutunk megoldáshoz.

e.3. A 0. megoldás vázlata

```
VanK:=Eldöntés(N,X,Megszámolás(.,X,T)=K)
```

Máris látszik a megoldás folytatásának a nehézsége: nem emelhető ki a Megszámolás tétel, hiszen az egyik paramétere (a ponttal jelölt elemszám) az Eldöntés tétel „futóindexe”.

e.4. A 0. megoldás kifejtése

```
Konstans
  MaxN:Egész(???)
Típus
  THk=Tömb(1..MaxN:TH)
  TTulFv=Függvény(Konstans x:TH):Logikai
…
```

```
        i:=1
        Ciklus amíg i≤N és nem Megszámolás(i,X,T)=K
α         i:+1
        Ciklus vége
        VanK:=i≤N
```

```
        …
        Függvény Megszámolás(Konstans n:Egész, x:THk, t:TTulFv):Egész
          db:=0
          Ciklus j=1-től n-ig
β           Ha t(x(j)) akkor db:+1
          Ciklus vége
          Megszámolás:=db
        Függvény vége.
```

Ahhoz, hogy bármit is tehessünk azért, hogy a megszámolás tétel (β) ne újra meg újra kezdődjék az eldöntés (α) ciklusban, meg kell szüntessük a függvény voltát, amihez legelőször ki kell transzformálnunk a feltételből. Ezt megtehetjük a PT12-vel:

```
        …
        i:=0; VanK:=Hamis
        Ciklus amíg i<N és nem VanK
α         i:+1
          VanK:=Megszámolás(i,X,T)=K
        Ciklus vége
        …
```

Ezzel még nem vagyunk elégedettek. A PT1-gyel elemeire bontjuk a `VanK`-ra vonatkozó értékadás jobboldalán lévő összetett kifejezést (`F(x):Megszámolás(x)`, `G(y):y=K`):

```
        …
        i:=0; VanK:=Hamis
        Ciklus amíg i<N és nem VanK
          i:+1
α         db:=Megszámolás(i,X,T)
          VanK:=db=K
        Ciklus vége
        …
```

A szétbontás közben ügyeltünk arra, hogy olyan segédváltozót vezessünk be (`db`), amely nem interferálhat az eldöntés algoritmusával.

Most már a `Megszámolás` függvény törzsét beilleszthetjük a megfelelő helyre. Ügyelnünk kell persze a paraméterek aktualizálása (`Megszámolás(i⇒N,X⇒X,T⇒T)`) közben arra, hogy a tágabb környezet adataival ne kerüljünk interferenciába. Így a törzsbeli ciklusváltozó `i`-jéből `j` (`i⇒j`) lesz, a „lokális” `db`-t megtarthatjuk, mivel éppen ez az azonosítója az eredeti függvény értékét befogadó `db`-nek (`db⇒db`).

```
        …
        i:=0; VanK:=Hamis
        Ciklus amíg i<N és nem VanK
          i:+1
          db:=0
α+β       Ciklus j=1-től i-ig
            Ha T(X(j)) akkor db:+1
          Ciklus vége
          VanK:=db=K
        Ciklus vége
        …
```

Formális bizonyítás nélkül is nyilvánvaló (elegendő a ciklusinvariánsra hivatkozni), hogy az `i`-s ciklussal eggyel előbbre jutva a megszámolás újraszámítása fölösleges, hiszen az `i-1`-re előbb kapott `db`-re újból ugyanazt kapnánk, tehát rövidíthető így:

α+β

```
…
i:=0; VanK:=Hamis; db:=0
Ciklus amíg i<N és nem VanK
  i:+1
  Ha T(X(i)) akkor db:+1
  VanK:=db=K
Ciklus vége
…
```

Magyarázat: a ciklus megszüntetése természetesen azzal jár, hogy a `db` inicializálását meg kellett szüntetni az `i`-s ciklusban, így viszont a `db`-növelés nem definiált értékre vonatkozna. Ezért a `db` inicializálást a ciklusba lépés előtt el kell végezni.

A PT12' fordított alkalmazásával további egyszerűsítést lehet véghez vinni:

α+β

```
…
i:=1; db:=0
Ciklus amíg i≤N és nem db=K
  Ha T(X(i)) akkor db:+1
  i:+1
Ciklus vége
VanK:=i≤N
…
```

Nem különösebben nehéz belátni, hogy az így kapott megoldás lényegében ekvivalens az előbbi, intuitíve kapott megoldással.

Az f) feladat megoldása

A sorozat legnagyobbika nagyobb-e, mint egy előre adott érték?
Gyakorlásul legyen ez is házi feladat.

Hivatkozások

[1] Szlávi P.: Programozási tételek specifikálása.
http://people.inf.elte.hu/szlavi/ProgModsz/Prtetel.pdf

[2] Szlávi P.: Programozási tételek egymásra építése * Programtranszformációk.
http://people.inf.elte.hu/szlavi/ProgModsz/Progtran.pdf

[3] Szlávi P.: Programok, programspecifikációk.
http://people.inf.elte.hu/szlavi/ProgModsz/Progspec.pdf

[4] Szlávi P.: Formális módszerek a programozásban.
http://people.inf.elte.hu/szlavi/PrM4felev/FormModsz.ppt