12.

Programtranszformációk

A továbbiakban olyan esetekkel fogunk foglalkozni, amelyekben a feladat állapotterét kell megváltoztatnunk. Vegyük észre: eddig is csináltunk már ilyet: bevezethetünk új változókat, vagy akár el is hagyhatunk komponenseket.

Az összetettebb esetekben az eredeti állapottér bizonyos komponenseit újakkal helyettesítjük.

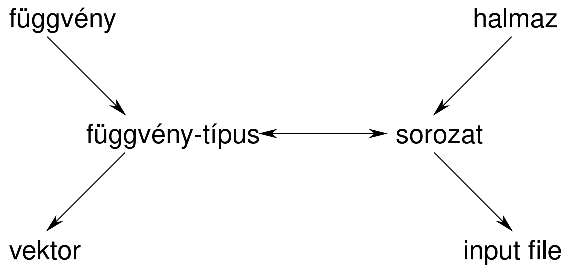
12.1. Koordináta transzformációk

Gyakran előfordul, hogy az állapottér egy komponensének típusát egy kapcsolódó másik típusra kell cserélnünk. Az ilyen transzformációk úgy általánosíthatók, hogy egy leképezést definiálunk a két állapottér egymásnak megfelelő komponensei között.

12.1.1. Típustranszformációk

Típustranszformációról akkor beszélhetünk, ha az állapottér bizonyos komponenseit valami kapcsolódó típusra cseréljük.

A programozási tételek – és általában véve a (feladat, megoldó program) párok – az alábbiakban bemutatásra kerülő transzformációkon keresztül általánosíthatók, ha az állapotterek közötti transzformáció tulajdonképpen típustranszformáció. Ekkor az ábrán látható valamely típuson megoldott program egyszerű szabályok segítségével átvihető egy kapcsolódó típusra.



12.1. ábra. Típusok közötti kapcsolatok

A programozási tételek átírása más típusra

Az alábbi sémák a programozási tételek típusok közötti transzformációjakor elvégzendő helyettesítéseket definiálják. A transzformáció úgy történik, hogy az első típus adott műveletét a második típus megfelelő (azonos sorban levő) műveletére cseréljük.

- **Halmazról** (x, y) **sorozatra** (a, b) (egy input halmaz/sorozat esetén)

  A két típus közötti megfeleltetés: az a és b sorozatok tagjai felsorolják az x és y halmazok elemeit.

$y := \emptyset$	$b := \langle\rangle$
$x \neq \emptyset$	$a.dom \neq 0$
e	$a.lov$
$y \tilde{\cup}$	$b : hiext^*$
$x \simeq$	$a : lorem$

  A fenti megfeleltetésben szereplő $hiext^*$ művelet a $hiext$ művelet olyan kiterjesztése, amely megengedi, hogy egy legfeljebb egy elemű halmazzal terjesszük ki a sorozatot. Ha a halmaz egy elemű, akkor azzal az elemmel terjeszti ki a sorozatot, míg ha üres, akkor a sorozatot helyben hagyja. Ennek megfelelően az elemenként feldolgozható f függvényről is feltesszük, hogy egy egy elemű halmazhoz legfeljebb egy elemű halmazt rendel hozzá. Az $e :\in x$ művelet helyett az e változónak feleltettük meg az $a.lov$ értéket, mert a lov függvény mindig a sorozat első elemét adja vissza ellentétben az eredeti értékkiválasztással.

  Példa (egyváltozós egyértékű elemenkénti feldolgozás):

$y := \emptyset$
$x \neq \emptyset$
$e :\in x$
$y := y \,\tilde{\cup}\, f(\{e\})$
$x := x \simeq e$

$b := \langle\rangle$
$a.dom \neq 0$
$b : hiext(f(\{a.lov\}))$
$a : lorem$

- **Sorozatról** (a) **szekvenciális input file-ra** (lopop,ext) (x) vagy (read) (y): (előreolvasási technika)

  A két típus közötti megfeleltetés: ha az a sorozat nem üres, akkor megegyezik a $loext(x, dx)$ és $loext(y, dy)$ sorozatokkal, míg üres sorozat esetén az x és y sorozatokkal.

	$dx, x : lopop$	$sy, dy, y : read$
$a.lov$	dx	dy
$a : lorem$	$dx, x : lopop$	$sy, dy, y : read$
$a.dom \neq 0$	$dx \neq extr$	$sy = norm$

  A transzformált program egy plusz $lopop$ (vagy $read$) művelettel kezdődik.

  Példa (egyváltozós egyértékű elemenkénti feldolgozás):

$b := \langle\rangle$
$a.dom \neq 0$
$e := a.lov$
$b : hiext(f(\{e\}))$
$a : lorem$

$b := \langle\rangle$
$sx, dx, x : read$
$sx = norm$
$e := dx$
$b : hiext(f(\{e\}))$
$sx, dx, x : read$

- **Halmazról** (x, y) **vektorra** $(v_1, v_2,\ i \in [v_1.lob, v_1.hib], j \in [v_2.lob, v_2.hib])$

  Feleltessük meg az x (és y) halmaznak a (v_1, i) (ill. (v_2, j)) párt oly módon, hogy az x (ill. y) halmaz elemei a v_1 vektor $v_1.lob..i$ (ill. a v_2 vektor $v_2.lob..j$) indexű elemeivel egyeznek meg.

$x \neq \emptyset$	$i \neq v_1.lob - 1$
e	$v_1[i]$
$x := x \simeq e$	$i := i - 1$
$y := y \,\tilde{\cup}\, d$	$v_2[j+1], j := d, j+1$
$y := \emptyset$	$j := v_2.lob - 1$

  Példa (egyváltozós egyértékű elemenkénti feldolgozás):

$y := \emptyset$
$x \neq \emptyset$
$\quad e :\in x$
$\quad y := y \,\tilde{\cup}\, f(\{e\})$
$\quad x := x \simeq \{e\}$

$j := v_2.lob - 1$
$i \neq v_1.lob - 1$
$\quad v_2[j+1], j := f(\{v_1[i]\}), j+1$
$\quad i := i - 1$

- **Sorozatról** (a, b) **vektorra** $(v_1, v_2,\ i \in [v_1.lob, v_1.hib]\ , j \in [v_2.lob, v_2.hib])$

  Sorozatok esetén a sorozat és vektor típusokra megengedett műveletek korlátozzák a két típus közötti megfeleltetés szabad választását: nem használható ugyanaz a leképezés, mint halmazoknál. Válasszunk hát a típusműveletekhez illeszkedő megfeleltetést!

  Feleltessük meg az a sorozatnak a (v_1, i) párt oly módon, hogy az a sorozat tagjai rendre a v_1 vektor $i..v_1.hib$ indexű elemeivel egyeznek meg. A b sorozatnak viszont a (v_2, j) párt úgy feleltessük meg, hogy a b sorozat elemei rendre a v_2 vektor $v_2.lob..j$ indexű elemeivel egyeznek meg.

$a.dom = 0$	$i = v_1.hib + 1$
$a.lov$	$v_1[i]$
$a : lorem$	$i := i + 1$
$b : hiext(e)$	$v_2[j+1], j := e, j+1$
$b := \langle\rangle$	$j := v_2.lob - 1$

  Példa (egyváltozós egyértékű elemenkénti feldolgozás):

$b := \langle\rangle$
$a.dom \neq 0$
$\quad b : hiext(f(\{a.lov\}))$
$\quad a : lorem$

$j := v_2.lob - 1$
$i \neq v_1.hib + 1$
$\quad v_2[j+1], j := f(\{v_1[i]\}), j+1$
$\quad i := i + 1$

- **Intervallum felett definiált függvényről** $([m, n]\ ,\ i \in [m, n]$ és a függvény argumentuma $i + 1)$ **sorozatra** (a)

  Első ránézésre úgy tűnhet, hogy kihagytunk egy lépést, a függvénytípust. De vegyük észre, hogy a függvény típusra való áttérés nem igényel transzformációt!

  Itt tulajdonképpen a sorozatot az intervallumnak feleltetjük meg az alábbi módon: tekintsük az a sorozatot egy az $[1..a.dom]$ intervallumon értelmezett függvénynek, ahol a függvényértékek a sorozat megfelelő elemei. Ekkor a tételben szereplő függvénynek tulajdonképpen az $f \circ a$ függvényt tekinthetjük.

$i \neq n$
$f(i+1)$
$i := i+1$

$a.dom \neq 0$
$f(a.lov)$
$a : lorem$

Példa (számlálás tétele):

$k, d := m-1, 0$		
$k \neq n$		
	$\beta(k+1)$	
	$d := d+1$	$SKIP$
	$k := k+1$	

$d := 0$		
$a.dom \neq 0$		
	$\beta(a.lov)$	
	$d := d+1$	$SKIP$
	$a : lorem$	

- **Halmazokról** (x, y, z) szigorúan monoton növő **sorozatokra** (a, b, c) a kétváltozós elemenkénti feldolgozás esetén. A kétváltozós elemenkénti feldolgozásnál az okozhat gondot, hogy el kell tudnunk dönteni, hogy egy adott elem melyik sorozatban van benne. Ezért tettük fel, hogy a sorozatok növekvően rendezettek, mert így mindkét sorozat legelső eleme a legkisebb, és ha ezek közül a kisebbiket választjuk, akkor a tartalmazás egyszerűen eldönthető. Egyébként a transzformáció többi része megegyezik az egyváltozós esetnél leírtakkal (itt is szükséges, hogy a függvényérték legfeljebb egyelemű legyen). Természetesen az elem kiválasztása itt is elhagyható, hiszen a lov művelet a sorozatnak mindig ugyanazt az elemét adja vissza.

$c := \langle\rangle$		
$a.dom \neq 0 \vee b.dom \neq 0$		
$(a.dom \neq 0 \wedge b.dom \neq 0 \wedge$ $a.lov < b.lov) \vee b.dom = 0$	$a.dom \neq 0 \wedge b.dom \neq 0 \wedge$ $a.lov = b.lov$	$(a.dom \neq 0 \wedge b.dom \neq 0 \wedge$ $a.lov > b.lov) \vee a.dom = 0$
$c : hiext(f(\{a.lov\}, \emptyset))$ $a : lorem$	$c : hiext(f(\{a.lov\},$ $\{b.lov\}))$ $a : lorem$ $b : lorem$	$c : hiext(f(\emptyset, \{b.lov\}))$ $b : lorem$

12.2. Állapottér transzformáció

Az alábbiakban egy olyan példát mutatunk be, amelyben az állapottér transzformáció nem egyszerű típustranszformáció. A feladatot úgy fogjuk megoldani, hogy eredeti állapottér helyett egy új (absztrakt) állapotteret választunk, azon megoldjuk a feladatot, majd a megoldásban szereplő műveleteket megvalósítjuk az eredeti téren.

Tegyük fel, hogy van egy karakterekből álló szekvenciális file-unk, ami egy szöveget tartalmaz. A feladat az, hogy számoljuk meg, hogy hány olyan szó van a szövegben, amelynek hossza nagyobb, mint a megadott k érték! A szövegben a szavakat elválasztó jelek (egy vagy több) választják el egymástól. Az elválasztó jelek halmazát jelölje S.

Így első ránézésre a feladat nem vezethető vissza egyetlen ismert programozási tételre sem. A feladatot ezért úgy általánosítjuk, hogy bevezetünk egy új állapotteret: tegyük fel, hogy a szövegfile helyett egy olyan file-unk van, amiben az eredeti file szavainak hossza szerepel. Legyen ez az absztrakt file $g : G$, míg az eredeti szövegfile $f : F$. A feladat specifikációja az új téren:

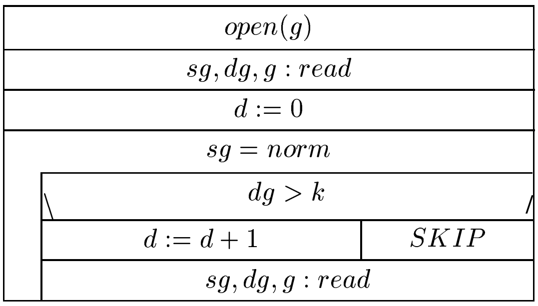
$$A = \underset{g}{G} \times \underset{d}{\mathbb{Z}}$$

$$B = \underset{g'}{G}$$

$$Q : (g = g')$$

$$R : (d = \sum_{i=1}^{dom(g)} \chi(g_i > k))$$

Ez a feladat visszavezethető a számlálás tételének szekvenciális file-ra felírt változatára:

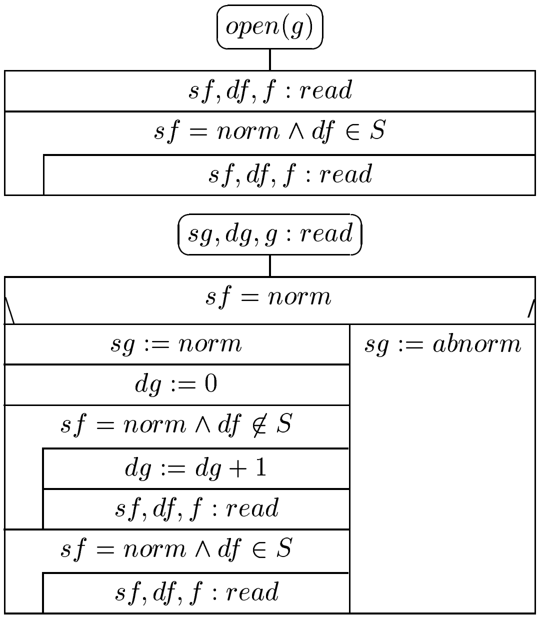


Hátra van még az absztrakt $open$ és $read$ műveletek megvalósítása.

Az absztrakt g file-nak pontosan akkor van még eleme, ha az eredeti f file-ban van még szó. Ezért van szükség az $open$ műveletre, amely arra hivatott, hogy biztosítsa a $read$ művelet elején megkívánt invariáns tulajdonságot: vagy $sf = abnorm$ vagy df a következő szó első karakterét tartalmazza. Ezen invariáns tulajdonság segítségével a $read$ műveletben könnyen el lehet dönteni, hogy lesz-e visszaadott érték, vagy már az absztrakt file is kiürült.

Az absztrakt file-ok kezelésének általában is ez a technikája: egy $open$ művelettel biztosítjuk a $read$ elején megkívánt invariáns tulajdonságot, és gondoskodunk róla, hogy a $read$ művelet ezt az invariáns tulajdonságot megtartsa. Általában is igaz az, hogy az invariánsból egyszerűen eldönthető, hogy lesz-e még visszaadott elem, ezért *az absztrakt $read$ mindig elágazással kezdődik!*

Nézzük tehát az absztrakt műveletek megvalósítását az eredeti állapottéren: ezek mint látható szintén programozási tételek egyszerű konstrukciói.



12.3. Egyszerű programtranszformációk

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

$$\forall a, b \in \mathcal{D}_{p(S)} : (\forall k \in ([1, n] \setminus \{i\} : a_k = b_k) \to p(S)(a) = p(S)(b).$$

Azt mondjuk, hogy az S program $a_i : A_i$ változója konstans, ha

$$\forall a \in A : \forall \alpha \in S(a) : (\forall k \in \mathcal{D}_\alpha : \alpha_{k\,i} = a_i).$$

Azt mondjuk, hogy az S program végrehajtása nem változtatja meg az $a_i : A_i$ változót, ha

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

Nem megengedett feltétel kitranszformálása elágazásból

Legyen $S \subseteq A \times A^{**}$, $S = IF(\pi_1 : S_1, ..., \pi_n : S_n)$, $A = A_1 \times ... \times A_m$. Konstruáljuk az $S' \subseteq A' \times A'^{**}$ programot az alábbi módon:

$$\begin{array}{ccccccccccc} A' & = & A_1 & \times & \dots & \times & A_m & \times & \mathbb{L} & \times \dots \times & \mathbb{L} \\ & & a_1 & & & & a_m & & l_1 & & l_n \end{array}$$

$l_1, \dots, l_n := \pi_1(a_1, ..., a_m), \dots, \pi_n(a_1, ..., a_m)$
$IF(l_1 : S_1, ..., l_n : S_n)$

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

Nem megengedett ciklusfeltétel kitranszformálása

Let $S \subseteq A \times A^{**}, S = DO(\pi : S_0)$, $A = A_1 \times ... \times A_m$. Konstruáljuk az $S' \subseteq A' \times A'^{**}$ programot az alábbi módon:

$$\begin{array}{ccccccccc} A' & = & A_1 & \times & \ldots & \times & A_m & \times & \mathbb{L} \\ & & a_1 & & & & a_m & & l \end{array}$$

$l := \pi(a_1, ..., a_m)$	
l	
	S_0
	$l := \pi(a_1, ..., a_m)$

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

Szimultán értékadás helyettesítése egyszerű értékadásokkal

Legyen $S \subseteq A \times A^{**}$, a következő szimultán értékadás:

$$\begin{array}{ccccccc} A & = & A_1 & \times & \ldots & \times & A_m \\ & & a_1 & & & & a_m \end{array}$$

S

$a_1, ..., a_m := f_1(a_1, ..., a_m), ..., f_m(a_1, ..., a_m)$

Konstruáljuk az $S' \subseteq A' \times A'^{**}$ programot az alábbi módon:

$$\begin{array}{ccccccccccccc} A' & = & A_1 & \times & \ldots & \times & A_m & \times & A_1 & \times & \ldots & \times & A_m \\ & & a_1 & & & & a_m & & b_1 & & & & b_m \end{array}$$

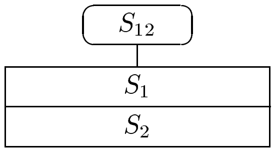
S'

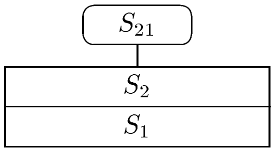
$b_1 := f_1(a_1, ..., a_m)$ $\vdots$ $b_m := f_m(a_1, ..., a_m)$
$a_1 := b_1$ $\vdots$ $a_m := b_m$

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

Szekvencia sorrendjének felcserélése

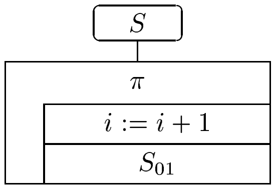
Ha $S_{12} = (S_1; S_2), S_{21} = (S_2; S_1)$ és az állapottér azon komponenseit amelyektől az S_1 függ S_2 nem változtatja meg és viszont, akkor S_{12} ekvivalens S_{21}-el.

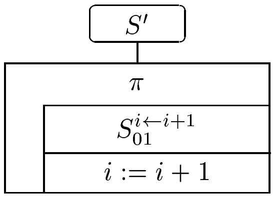




Ciklusmag-beli szekvencia sorrendjének felcserélése

Legyen $S = DO(\pi : S_0)$, $S_0 = (i := i + 1; S_{01})$ és tegyük fel, hogy az i konstans S_{01}-ben. Legyen továbbá $S' = ({S_{01}}^{i \leftarrow i+1}; i := i + 1)$.



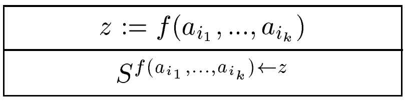


Ekkor S ekvivalens S'-vel.

Függvény helyettesítése változóval

Legyen $S \subseteq A \times A^{**}$, $A = A_1 \times ... \times A_m$. és legyen f egy az $A_{i_1} \times ... \times A_{i_k}$ altér felett 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:

$$\begin{array}{ccccccccc} A' & = & A_1 & \times & \dots & \times & A_m & \times & H \\ & & a_1 & & & & a_m & & z \end{array}$$



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, $t_0, t_{-1}, \dots, t_{-k+1} \in \mathbb{Z}$ rögzített, és definiáljuk az $f : \mathbb{Z} \to H$ 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:

<table>
<tr><td colspan="2">$i := 0$</td></tr>
<tr><td colspan="2">S_1</td></tr>
<tr><td colspan="2">π</td></tr>
<tr><td rowspan="2"></td><td>S_0</td></tr>
<tr><td>$i := i + 1$</td></tr>
</table>

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:

<table>
<tr><td colspan="2">$i, z, z_{-1}, .., z_{-k+1} := 0, t_0, t_{-1}, .., t_{-k+1}$</td></tr>
<tr><td colspan="2">S_1</td></tr>
<tr><td colspan="2">π</td></tr>
<tr><td rowspan="3"></td><td>$z, z_{-1}, .., z_{-k+1} := F(i+1, f(i), ..., f(i-k+1)), z, .., z_{-k+2}$</td></tr>
<tr><td>${S_0}^{f(i+1) \leftarrow z}$</td></tr>
<tr><td>$i := i + 1$</td></tr>
</table>