I. Állítsuk elő két halmaz metszetét!

Specifikáció

$A = (x{:}set(E), y{:}set(E), z{:}set(E))$

$Ef = (x = x' \wedge y = y')$

$Uf = (z = x' \cap y') = (z = \bigcup_{e \in x' \cup y'} f(e))$

ahol $f{:}E \rightarrow 2^F$ és

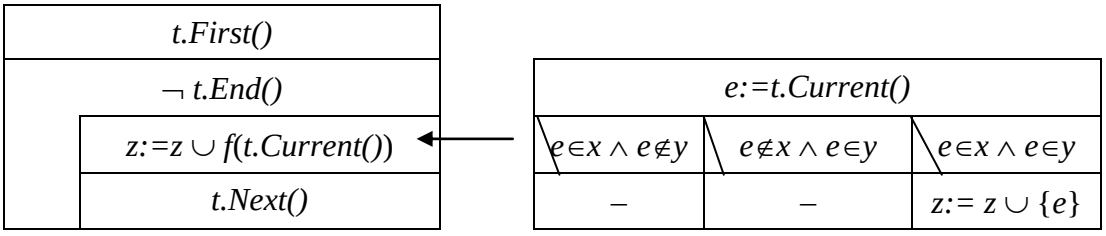
$$f(e) = \begin{cases} \emptyset & \text{ha} \quad e \in x' \wedge e \notin y' \\ \emptyset & \text{ha} \quad e \notin x' \wedge e \in y' \\ \{e\} & \text{ha} \quad e \in x' \wedge e \in y' \end{cases}$$

Összegzés

$e \in t$	~	$e \in x \cup y$ (halmaz elemeinek felsorolása)
$H, +, 0$	~	$\boldsymbol{2^E}, \cup, \emptyset$

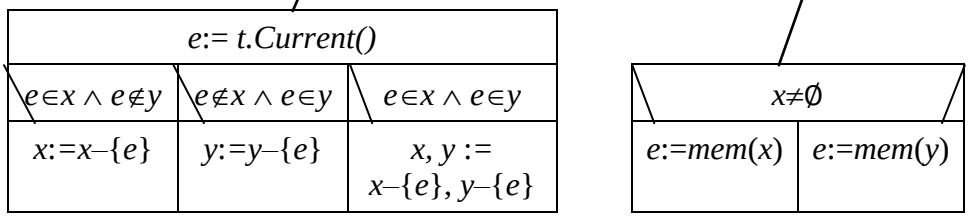
Algoritmus váz

A $z{:=}z \cup f(Current())$ részfeladat felbontható az $e{:=}Current()$ és a $z{:=}z \cup f(e)$ részfeladatok szekvenciájára, a $z{:=}z \cup f(e)$ részfeladat pedig egy elágazás lesz:



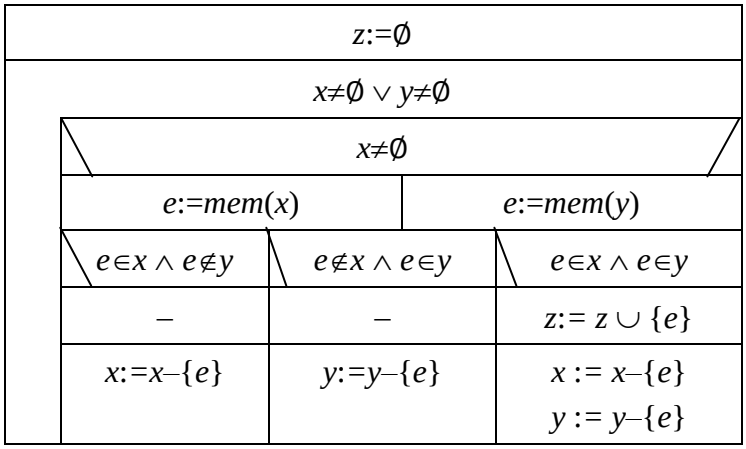
enor(E) (olyan, mintha az $x \cup y$ elemeit kellene felsorolni, amit közvetlenül nem lehet)

t:enor(E)	*t.First()*	*t.Next()*	*l:= t.End()*	*e:= t.Current()*
$x \cup y{:}set(E)$	–	$x \cup y := x \cup y – mem(x \cup y)$	$l := x \cup y = \emptyset$	$e := mem(x \cup y)$
$x{:}set(E)$ $y{:}set(E)$	–		$l := x = \emptyset \wedge y = \emptyset$	



Algoritmus egyben

A $z:=z\cup f(e)$ és a *Next()* elágazásai egybevonhatók:

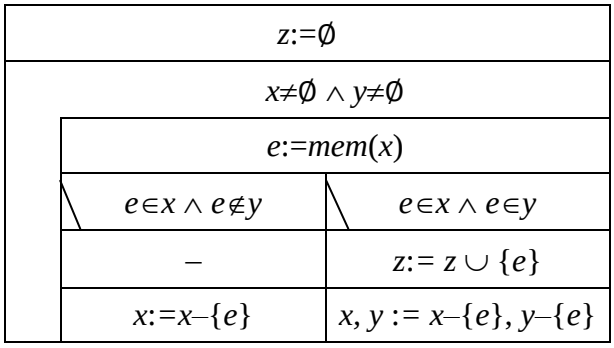


Egyszerűsíthető a fenti program:

l:= t.End() ~ $l:= x=\emptyset \vee y=\emptyset$

e:=*t.Current()* ~ *e*:=*mem*(*x*)

t.Next() ~ egyik ága felesleges.



Megjegyzés: Gondoljuk át, mi lenne a megoldás, a két halmaz különbségét, esetleg szimmetrikus differenciáját vagy unióját kell előállítani.

II. Dolgozzuk fel két rendezett és egyértelmű felsorolás elemeit!

Specifikáció

$A = (x{:}enor(E), y{:}enor(E), z{:}\ F^*)$ ahol $f{:}E \rightarrow F^*$ és

$Ef = (x{=}x' \wedge y{=}y' \wedge x\uparrow \wedge y\uparrow)$

$Uf = (z = \underset{e \in x' \cup y'}{\oplus} f(e))$

$$f(e) = \begin{cases} f_1(e) & ha \quad e \in \{x'\} \wedge e \notin \{y'\} \\ f_2(e) & ha \quad e \notin \{x'\} \wedge e \in \{y'\} \\ f_3(e) & ha \quad e \in \{x'\} \wedge e \in \{y'\} \end{cases}$$

és az $f_1, f_2, f_3{:}E \rightarrow F^*$ adott függvények

Az utófeltételben az $e \in x \cup y$ szimbólum arra utal, hogy az x és y felsorolásában szereplő elemeket kell felsorolni. Az $\{x\}$ az x felsorolás elemeinek halmazát jelöli.

Algoritmus váz

A megoldás az x és y elemeinek közös (összefuttatott) felsorolására ($t{:}enor(E)$) épített összegzés.

$z{:=}<>$ $t.First()$	
$\neg t.End()$	
	$z{:=}z \oplus f(t.Current())$
	$t.Next()$

A felsorolás során lényeges, hogy a *t.Current()* által visszaadott elemről meg tudjuk mondani, hogy az csak az *x*, csak az *y*, vagy mindkét felsorolásban szerepel-e.

Összefuttatott felsoroló: enor(E)

E^*	*t.First()*	*t.Next()*	$l{:=}$ *t.End()*	$e{:=}$ *t.Current()*
x: enor(E) *y: enor(E)*	*x.First()* *y.First()*	lásd később	$l{:=}$ *x.End()* $\wedge$ *y.End()*	lásd később

Nyilvánvaló, hogy az összefuttatás a két felsorolóra épül: azokkal reprezentáljuk az összefuttatott felsorolót. Az összefuttatáshoz kezdetben mindkét felsorolást el kell indítani (lásd *First()*), és akkor áll le, ha mindkettő befejeződött (lásd *End()*). Kihasználva, hogy külön-külön mindkét felsorolás szigorúan növekedően rendezett, az alábbi elágazással mindig kiválaszthatunk olyan elemet (*e*), amelyről megmondható, hogy az csak az *x*, csak az *y*, vagy mindkét felsorolásnak eleme-e.

e:=Current()

y.End() $\vee$ ($\neg$*x.End()* $\wedge$ *x.Current()*<*y.Current()*)	*x.End()* $\vee$ ($\neg$*y.End()* $\wedge$ *x.Current()*>*y.Current()*)	$\neg$*x.End()* $\wedge$ $\neg$*y.End()* $\wedge$ *x.Current()*=*y.Current()*
e:=*x.Current()*	*e*:=*y.Current()*	*e*:=*x.Current()*

A *Next()* műveletnek a *Current()* műveletben kiválasztott elemet kell „kivennie” mindkét felsorolásból, azaz attól függően, hogy a kiválasztott elem csak az *x*, csak az *y*, vagy mindkét felsorolásban szerepel, a megfelelő felsorolásokban tovább kell lépni.

t.Next()

y.End() ∨ (¬*x.End()* ∧ *x.Current()*<*y.Current()*)	*x.End()* ∨ (¬*y.End()* ∧ *x.Current()*>*y.Current()*)	¬*x.End()* ∧ ¬*y.End()* ∧ *x.Current()*=*y.Current()*
x.Next()	*y.Next()*	*x. Next(); y.Next()*

Megjegyzés: Az összefuttatás az $\{x\}\cup\{y\}$ elemeit rendezettségük sorrendjében sorolja fel.

Főprogram

A $z:=z \oplus f(t.Current())$ értékadást az $e:=Current()$ és a $z:=z\oplus f(e)$ összevonásával kapjuk:

y.End() ∨ (¬*x.End()* ∧ *x.Current()*<*y.Current()*)	*x.End()* ∨ (¬*y.End()* ∧ *x.Current()*>*y.Current()*)	¬*x.End()* ∧ ¬*y.End()* ∧ *x.Current()*=*y.Current()*
$z:= z \oplus f_1(x.Current())$	$z:= z \oplus f_2(y.Current())$	$z:= z \oplus f_3(x.Current())$

Ezt összevonhatjuk a *t.Next()* elágazásával, és összességében az alábbi programhoz jutunk.

z:=<>; *x.First(); y.First()*		
¬*x.End()* ∨ ¬*y.End()*		
y.End() ∨ (¬*x.End()* ∧ *x.Current()*<*y.Current()*)	*x.End()* ∨ (¬*y.End()* ∧ *x.Current()*>*y.Current()*)	¬*x.End()* ∧ ¬*y.End()* ∧ *x.Current()*=*y.Current()*
$z:= z \oplus f_1(x.Current())$	$z:= z \oplus f_2(x.Current())$	$z:= z \oplus f_3(x.Current())$
x.Next()	*y.Next()*	*x. Next(); y.Next()*

Megjegyzés

Az *f* függvény ismeretében egyszerűsíthető a fenti program. Ha például csak a közös elemeket kell változtatás nélkül kigyűjteni, akkor egyszerűsödik a ciklusfeltétel (¬*x.End()* ∧ ¬*y.End()*), és emiatt az elágazás feltételek is.

z:=<>; *x.First(); y.First()*		
¬*x.End()* ∧ ¬*y.End()*		
x.Current()<*y.Current()*	*x.Current()*>*y.Current()*	*x.Current()*=*y.Current()*
–	–	$z:= z \oplus <x.Current()>$
x.Next()	*y.Next()*	*x. Next(); y.Next()*

Alkalmazások:

a) Kik voltak azok az alapító atyák közül származó USA elnökök, akik nem voltak a virginiai klán tagjai?

$A = (x{:}infile(String), y{:}infile(String), z{:}outfile(String))$
$Ef = (x{=}x' \wedge y{=}y' \wedge x\uparrow \wedge y\uparrow)$
$Uf = (z = \underset{e \in x' \cup y'}{\oplus} f(e))$

ahol $f{:}String \rightarrow String^*$ és

$$f(e) = \begin{cases} <e> & ha \quad e \in \{x'\} \wedge e \notin \{y'\} \\ <> & ha \quad e \notin \{x'\} \wedge e \in \{y'\} \\ <> & ha \quad e \in \{x'\} \wedge e \in \{y'\} \end{cases}$$

$z := <>$ *sx,dx,x:read; sy,dy,y:read;*			
sx=norm ∨ sy=norm			~~*sy=norm*~~
sy=abnorm ∨ (sx=norm ∧ dx<dy)	*sx=abnorm ∨ (sy=norm ∧ dx>dy)*	*sx=norm ∧ sy=norm ∧ dx=dy*	~~*sx=norm*~~ ~~*sx=abnorm*~~
z:write(dx)	–	–	
sx,dx,x:read	*sy,dy,y:read*	*sx,dx,x:read;* *sy,dy,y:read*	

b) Hány olyan elnöke volt az USA-nak, aki az alapító atyák közé tartozott vagy a virginiai klán tagja volt, és a nevében hétnél kevesebb betű van?

$A = (x{:}infile(String), y{:}\ infile(String), s{:}\mathbb{N})$
$Ef = (x{=}x' \wedge y{=}y' \wedge x\uparrow \wedge y\uparrow)$
$Uf = (s = \underset{\substack{e \in x' \cup y' \\ |e|<7}}{\sum} 1 \ )$

s := 0 *sx,dx,x:read; sy,dy,y:read;*		
sx=norm ∨ sy=norm		
sy=abnorm ∨ (sx=norm ∧ dx<dy)	*sx=abnorm ∨ (sy=norm ∧ dx>dy)*	*sx=norm ∧ sy=norm ∧ dx=dy*
|dx| < 7	*|dy| < 7*	*|dx| < 7*
s:=s+1 | –	*s:=s+1* | –	*s:=s+1* | –
sx,dx,x:read	*sy,dy,y:read*	*sx,dx,x:read;* *sy,dy,y:read*

c) Keressünk olyan USA elnököt, aki vagy az alapító atyák közé tartozott, de nem volt a virginiai klán tagja, vagy a virginiai klán tagja volt, de nem volt alapító atya!

$A = (x{:}infile(String), y{:}\ infile(String), l{:}\mathbb{L}, e{:}String)$
$Ef = (\ x{=}x' \wedge y{=}y' \wedge x\uparrow \wedge y\uparrow)$
$Uf = (\ l, elem = \underset{e \in x' \cup y'}{search}, e \in \{x'\} \circ \{y'\})$

$l := hamis$ $sx,dx,x:read;\ sy,dy,y:read;$			
$\neg l \ \wedge (sx{=}norm \vee sy{=}norm)$			
	$sy{=}abnorm \vee (sx{=}norm \wedge\ dx{<}dy)$	$sx{=}abnorm \vee (sy{=}norm \wedge\ dx{>}dy)$	$sx{=}norm \wedge sy{=}norm \wedge dx{=}dy$
	$l, elem{:=}igaz, dx$	$l, elem{:=}igaz, dy$	–
	$sx,dx,x:read$	$sy,dy,y:read$	$sx,dx,x:read;$ $sy,dy,y:read$