

Programazioaren Metodologia

May 12, 2015

ABIZENAK:
IZENA:
TITULAZIOA: Gradua
TALDEA: Euskara

(2 puntu) Ondoko funtzioren espezifikazio ekuazionala idatzi:

`agerpen-pos: pila(T) × T × positive → bool`

p pila bat, x elementu bat eta n zenbaki positiboa emanda, `agerpen-pos` funtziok p-ren n-garren posizioan (gailurretik behera kontatuta) x-a agertzen den ala ez itzultzen du.

$$p = \begin{array}{|c|} \hline 4 \\ \hline 5 \\ \hline 6 \\ \hline 3 \\ \hline \end{array} \quad x = 6 \quad n = 3 \quad \Rightarrow \quad \text{agerpen-pos}(p, x, n) = \text{true}$$

datu-mota: `pila(T)`

eragiketa: `agerpen-pos : pila(T) × T × positive → bool`

ekuazioak:

EBAZPEN1

(1) `agerpen-pos(pilahutsa, x, n) = errore`

(2) $\text{agerpen-pos}(\text{pilaratu}(p, y), x, 1) = \begin{cases} \text{true} & \text{baldin } y = x \\ \text{false} & \text{bestela} \end{cases}$

(3) `agerpen-pos(pilaratu(p, y), x, n + 1) = agerpen-pos(p, x, n)`

EBAZPEN2

(1) `agerpen-pos(pilahutsa, x, n) = errore`

(2) $\text{agerpen-pos}(\text{pilaratu}(p, y), x, n) = \begin{cases} \text{true} & \text{baldin } y = x \wedge n = 1 \\ \text{false} & \text{baldin } y \neq x \wedge n = 1 \\ \text{agerpen-pos}(p, x, n - 1) & \text{bestela} \end{cases}$

(2.5 puntutu) Bustallen metodoa erabiliz, discimatu algoritmo iteratibo bat aurritzki funtzioa konputatzuen duena.

$\text{aurritzki} : \text{sekuentzia(T)} \times \text{sekuentzia(T)} \rightarrow \text{boolean}$

Especifikazio ekuaizionala:

- (1) $\text{aurritzki}(\langle \rangle, s) = \text{true}$
 - (2) $\text{aurritzki}(x \bullet s, \langle \rangle) = \text{false}$
 - (3) $\text{aurritzki}(x \bullet s, y \bullet r) = (x = y) \wedge \text{aurritzki}(s, r)$
- Funtzio errekutsiboa gisa definituta:
- ```

function aurritzki (s,r:sekuentzia(T)) return b:boolean is
 if hutsa.da(s) then b := true;
 elsif hutsa.da(r) then b := false;
 else b := (lehena(s) = lehena(r)) and aurritzki(hondarra(s),hondarra(r));
 end if;

```
- function aurritzki (s,r:sekuentzia(T)) return b:boolean is
 if hutsa.da(s) then b := true;
 elsif hutsa.da(r) then b := false;
 else b := (lehena(s) = lehena(r)) and aurritzki(hondarra(s),hondarra(r));
 end if;

**INBARIANTEA/ERREKURRENTZI ERLAZIOA :**

Aurre  $\equiv \{\text{True}\}$

Post  $\equiv \{z \leftrightarrow \text{aurritzki}(s, r)\}$

Inb  $\equiv \{ \text{aurritzki}(s, r) = b \wedge \text{aurritzki}(u, v) \}$

KASU 1:  $\text{hutsa.da}(u) \Rightarrow \text{aurritzki}(s, r) = b \wedge \text{True} = b$

KASU 2:  $\neg \text{hutsa.da}(u) \wedge \text{hutsa.da}(v) \Rightarrow \text{aurritzki}(s, r) = b \wedge \text{False} = \text{False}$

**HASIERAKETA :**  $(b, u, v) := (\text{True}, s, r)$

$\text{aurritzki}(s, r) = b \wedge \text{aurritzki}(u, v) = \text{True} \wedge \text{aurritzki}(s, r) = \text{aurritzki}(s, r)$

**BUKAERA :**  $\text{hutsa.da}(u) \vee \text{hutsa.da}(v)$

Beraz, bukaera honako hau da:

if hutsa.da(u) then z := b; else z := False; end if

**ITERAZIOAREN GORPUTZA(TOLESKETA/DESTOLESKETA) :**

$\neg \text{hutsa.da}(u) \wedge \neg \text{hutsa.da}(v) \Rightarrow \text{aurritzki}(s, r) = b \wedge \text{aurritzki}(u, v) \bullet \text{hondarra}(u) \bullet \text{hondarra}(v) = b \wedge (\text{lehena}(u) = \text{lehena}(v) \wedge \text{aurritzki}(hondarra(u), hondarra(v)) = b)$

(b  $\wedge$  lehena(u) = lehena(v))  $\wedge$  aurritzki(hondarra(u), hondarra(v)) =  
 $\underbrace{(b \wedge lehena(u) = lehena(v))}_{(b \wedge lehena(u) = lehena(v)) \wedge aurritzki(hondarra(u), hondarra(v))} =$

$b' \quad \wedge \quad \text{aurritzki}(u', v')$

**FUNTZIO DOKUMENTATUA :**

```

function aurritzki.it (s,r:sekuentzia(T)) return z:boolean is
 u,v:sekuentzia(T); b:boolean;
 Aurre $\equiv \{\text{True}\}$
 b := True; u := s; v := r;
 INB $\equiv \{ \text{aurritzki}(s, r) = b \wedge \text{aurritzki}(u, v) \}$
 E = luzera(u)
 while not hutsa.da(u) and not hutsa.da(v) loop
 { aurritzki(s,r) = b \wedge aurritzki(lehena(u) \bullet hondarra(u) \bullet hondarra(v)) }
 b := b and (lehena(u) = lehena(v));
 { aurritzki(s,r) = b \wedge aurritzki(hondarra(u), hondarra(v)) }
 u := hondarra(u);
 { aurritzki(s,r) = b \wedge aurritzki(u, hondarra(v)) }
 v := hondarra(v);
 en loop;
 if hutsa.da(u)
 then { aurritzki(s,r) = b \wedge aurritzki(<>, v) }
 z := b;
 else { aurritzki(s,r) = b \wedge aurritzki(lehena(u) \bullet hondarra(u), <>) }
 z := False;
 end if
 Post $\equiv \{z \leftrightarrow \text{aurritzki}(s, r)\}$

```



$$\text{Behetv}(\text{Pilaratu}(P, x), e, \text{Pilaratu}(q, x)) = \\ = \begin{cases} \text{Behetv}(P, e, \text{Pilaratu}(q, x)) & \text{Baldin } x \leq e \\ \text{Behetv}(P, e, q) & \text{Bestela} \end{cases}$$

Kendu

Mota: Pila(t)

Funtzioa:  $\text{Pila}(t) \times t \rightarrow \text{Pila}(t)$

Ekuazioak:

Kendu(PilaHutse, x) = PilaHutse

$$\text{Kendu}(\text{Pilaratu}(P, x), e) = \begin{cases} \text{Pilaratu}(P, x) & \text{Baldin } e = x \\ \text{Kendu}(P, e) & \text{Bestela.} \end{cases}$$

4.6

Mota: Sekuentzia(t)

Funtzioa: Sekuentzia(t)  $\rightarrow$  Boolean

Ley: Boolean

Ekuazioak:

OndorOndollo( $\perp$ )  $\Rightarrow$  False

$$\text{OndorOndollo}(e, s) = \begin{cases} \text{true} & \text{Baldin } e = \text{berera}(s) \\ \text{OndorOndollo}(s) & \text{Bestela.} \end{cases}$$

Motiv: Schleventzia ( $\top$ )

Leg: Boolean

funktion:  $\top \times \text{Schleventzia}(\top) \rightarrow \text{Boolean}$

Ergebnisse:

Bedingo ( $n, \langle \rangle$ ) = false

Bedingo ( $n, e, s$ ) =  $\begin{cases} \text{true} & \text{Baldin, } n = e \\ \text{Bedingo}(n, s) & \text{Bestell} \end{cases}$

4.5

Motiv: Schleventzia ( $\top$ )

funktion: Schleventzia ( $\top$ )  $\rightarrow$  Schleventzia ( $\top$ )

Ergebnisse:

Esabeku Ecker ( $\langle \rangle$ ) = false

Esabeku Ecker ( $e, s$ ) =  $\begin{cases} \text{Eckbecker}(\top) & \text{Baldin \& Bedingo}(e, s) \\ e, \text{Esabeku Ecker}(s) & \text{Bestell} \end{cases}$

Bahetu

Motiv: Pila ( $\top$ )

funktion: Pila ( $\top$ )  $\times \top \times$  Pila ( $\top$ )  $\rightarrow$  Pila ( $\top$ )

Ergebnisse:

Bahetu (PilaHutse, e, PilaHutse) = PilaHutse

Bahetu (PilaHutse, e, Pilaratu ( $q, x_1$ )) = Pilaratu ( $q, x_1$ )

Bahetu (Pilaratu ( $P, x$ ), e, Pilaratu (PilaHutse))

=  $\begin{cases} \text{Bahetu} (P, e, \text{Pilaratu} (\text{PilaHutse}, x)) & \text{Baldin } x \leq e \\ \text{Bahetu} (P, e, \text{PilaHutse}) & \text{Bestell} \end{cases}$

(3)

3.1

Ez da zuzena, ez baita kontuan hartzen salutza denean.

3.2

Zuzena da, hala guztioak hortzen baitira kontuan.

3.3

Ez da zuzena, 3. puntuen n=0 dek eosten bedugu ~~ke emaitzean~~  
n=1 hortzen dugu, hor da -1 zu ordeiora, gizki dago planteak hitea.

(4)

4.2

Nota: Selventzia ( $t$ )

funtzioa: Selventzia ( $t$ )  $\rightarrow$  Selventzia ( $t$ )

Ekuazioak:

Inplementatu ( $\leftrightarrow$ ) =  $\leftrightarrow$

Inplementatu ( $e, s$ ) =  $(e+1) \oplus$  Implementatu ( $s$ )

4.3

Nota: Selventzia ( $t$ )

Lag: Boolean

funtzioa: Selventzia ( $t$ )  $\rightarrow$  Boolean

Ekuazioak:

Errepikaturik ( $\leftrightarrow$ ) = False

Errepikaturik ( $e, s$ ) =  $\begin{cases} \text{Errepikaturik } (s) & \text{Baldin } Bedago \text{ } (e, s) = \text{false} \\ \text{true} & \text{Baldin } Bedago \text{ } (e, s) = \text{true} \end{cases}$

Oraint Bedago funtzioa  $\Rightarrow$  errepikatzen dugu!

# PROGRAMAZIOAREN METODOLOGIA

## 2010eko ekainaren 10a

Sistemen Informatikarien Ingeniaritza / Informatikarien Ingeniaritza  
Donostiarra

- (1 puntu) Idatzi lehen mailako formula bat honako eruntzianu adierazteko:  
 $A(1..n)$  bektoean boda guxienetako elementu bat  $B(1..m)$  bektoean behin baino gehiagotan agertzen dena.
- (2 puntu) Frogatu formalki baitzapaten hau:

$$\exists k (1 \leq k \leq n \wedge Np(1 \leq p \leq m \wedge A(k) = B(p)) > 1)$$

- (2 puntu) Idatzi lehen mailako formula bat honako eruntzianu adierazteko:

$A(1..n)$  bektoean boda guxienetako elementu bat  $B(1..m)$  bektoean behin baino gehiagotan agertzen dena.

$$(1) \{n \geq 1\}$$

$$i := 1; s := 0;$$

$$(2) \text{ INB} = \left\{ \begin{array}{l} \forall k 1 \leq k < i \rightarrow B(k) = \frac{\sum_{j=1}^k A(j)}{k} \\ \wedge s = \sum_{j=1}^{i-1} A(j) \wedge 1 \leq i \leq n+1 \end{array} \right\}$$

$R = n + 1 - i$

while  $i \leq n$  loop

$$(3) \left\{ \begin{array}{l} \forall k 1 \leq k < i \rightarrow B(k) = \frac{\sum_{j=1}^k A(j)}{k} \\ \wedge s = \sum_{j=1}^{i-1} A(j) \wedge 1 \leq i \leq n \end{array} \right\}$$

$s := s + A(i);$

$$(4) \left\{ \begin{array}{l} \forall k 1 \leq k < i \rightarrow B(k) = \frac{\sum_{j=1}^k A(j)}{k} \\ \wedge s = \sum_{j=1}^i A(j) \wedge 1 \leq i \leq n \end{array} \right\}$$

$B(i) := s/i;$

$$(5) \left\{ \begin{array}{l} \forall k 1 \leq k \leq i \rightarrow B(k) = \frac{\sum_{j=1}^k A(j)}{k} \\ \wedge s = \sum_{j=1}^i A(j) \wedge 1 \leq i \leq n \end{array} \right\}$$

*Beldur bat uztse-ka (ezw) Rautzeta-ka*

$$\text{Ezutse-Seku : Abilt } (\tau) \rightarrow \text{Sekuen Vgaz}(\tau)$$

Ezutse-Seku :

$$\text{ezutse-Seku (Abilt)} = \left\{ \begin{array}{l} c > @(\text{ezutse-Seku (ezw)}) @(\text{ezutse-Seku (esk)}) \\ \quad i := i + 1; \end{array} \right.$$

Ezutse-Seku (ezw, esw) =

$$(6) \left\{ \begin{array}{l} \text{ezutse-Seku (esw)} @ \text{ezutse-Seku (ezw)} \\ \quad \left\{ \begin{array}{l} \forall k 1 \leq k \leq n \rightarrow B(k) = \frac{\sum_{j=1}^k A(j)}{k} \end{array} \right\} \end{array} \right\}$$

4. (2 puntu) bahetu funtziaren espezifikazio ekuzionala idatzi:

bahetu:  $pila(T) \times T \times pila(T) \rightarrow pila(T)$

$p$  pila bat,  $x$  elementu bat eta  $q$  pila bat emanda, bahetu funtzoak pila bat lortzen du.  
Emaitzan,  $q$  pilaren gainean  $x$  baino caixigaoak edo berdinak diren  $p$ -ko elementuak  
agertzen dira,  $p$ -n zeuden alderantzikako ordenan. Hau da, bahetu funtzoak  $p$  pilaren  
edukia irudi egiten du  $q$  pilaren gainean, baina  $x$  baino handiagoak direnak kenduz.

|       |   |   |     |   |
|-------|---|---|-----|---|
| $p =$ | 4 | 5 | 6   | 3 |
|       |   |   | x=4 | 7 |
|       |   |   |     | 9 |
|       |   |   |     | 8 |
|       |   |   | q=  |   |

$p, q \in Pila(T); x, e \in T$

(1)  $bahetu(pilahutsa, x, q) = q$

(2)  $bahetu(Pilaratu(p, e), x, q) = \begin{cases} bahetu(p, x, q), & baldin e > x \\ bahetu(p, x, Pilaratu(q, e)), & bestela \end{cases}$

5. (2,5 puntu) Burstableen metoda erabiliz, disenau algoritmo iteratibo bat banandu funtzia kalkulatzeko ducena.

banandu:  $pila(T) \rightarrow sekuentzia(T)$

Espezifikazio ekuzionala:

$$(1) banandu(Pilahutsa) = <>$$

$$(2) banandu(Pilaratu(p, x)) = \begin{cases} x \bullet banandu(p) & baldin bitxita(x) \\ banandu(p) @ < x > & baldin bakoitza(x) \end{cases}$$

Funtzio errekurriboko gisa definituta:

```
function banandu (p:Pila(T)) return s:sekuentzia (T) is
 if Hutsa_Da(p) then s := < >;
 elsif Bikoitia(Gailurrekoia(p)) then
 s := Gailurrekoia(p) • banandu (despilatu(p));
 else s := banandu (despilatu(p)) @ < Gailurrekoia(p) >;
 end if;
```

INBARIANTEA eta BORNE-ADIERAZPENA:

INB:  $banandu(P) = S @ banandu(R) @ T$

non:  $R \in Pila(T); S, T \in Sekuentzia(T)$

E = Altuera(R)

HASTERAKETA:

$S := <>; R := P; T := <>;$

```
banandu (P) = S @ banandu (R) @ T
 = <> @ banandu (P) @ <>
 = banandu (P)
```

## BUKAERAREN AZTERKETA:

### FUNTZIO ITERATIBOA:

```
function banandu_it (p:pila(T)) return Sekuentzia(T) is
 R: Pila(T);
 S,T: Sekuentzia(T);
begin
 S:=<>;
 R:=p;
 T:=<>;
 INB = { banandu(P) = S@banandu(R)@T }
```

- $Hutsa\_Da(R) \rightarrow banandu(P) = S @ banandu(pilahutsa) @ T$   
 $= S @ <> @ T$   
 $= S @ T$

## ITERAZIOAREN GORPUTZA:

- $Bikoitia(Gailurrekoia(R)) \rightarrow$ 
  - $banandu(P) = S @ banandu(R) @ T$   
 $= S @ (Gailurrekoia(R) \bullet banandu(Despilatu(R))) @ T$   
 $= (S @ <Gailurrekoia(R)>) @ banandu(Despilatu(R)) @ T$   
 $= S' @ banandu(R') @ T'$   
 $S := S @ <Gailurrekoia(R)>;$   
 $R := Despilatu(R);$
  - $not Bikoitia(Gailurrekoia(R)) \rightarrow$ 
    - $banandu(P) = S @ banandu(R) @ T$   
 $= S @ (banandu(Despilatu(R)) @ <Gailurrekoia(R)>) @ T$   
 $= S @ banandu(Despilatu(R)) @ (<Gailurrekoia(R)> @ T)$   
 $= S' @ banandu(R') @ T'$   
 $T := <Gailurrekoia(R)> @ T;$   
 $R := Despilatu(R);$
    - $end if;$   
 $end loop;$   
 $return S @ T;$   
 $end banandu\_it; \{ banandu\_it(p) = banandu(p) \}$



# 6

## Programa errekurtsiboen transformazioa: Burstall-en metodoa

### 6.1 Burstall-en metodoa, programa errekurtsiboak iteratibo bihurtzeko

Programa errekurtsiboak soluziorik *zuzenena*, *simpleena* eta *argiena* izaten dira zenbait problematarako. Esate baterako:

- Indukzio bidez definitutako funtzieak.
- Datu-mota induktiboen tratamendua.
- Izaera bereko azpiproblematan banatzen diren problemak.

Baina askotan, batez ere programa iteratiboekin konparatzu, ez dira eraginkorrenak. Eraginkor ez izate hori bi arrazoi hauengatik gertatzen da:

- Kalkuluen errepikapenagatik.
- Parametroak gorde eta berreskuratu behar direlako deiak kateatzean.

Hala ere, esan dugunez, zenbaitetan errazagoa da algoritmo errekurtsiboa garatzea iteratiboa garatzea baino. Horregatik, oso interesgarria da ebazpide errekurtsiboa abi-apuntu gisa erabiltzea programa iteratibo bat lortzeko, bereziki batetik besterako bihurketa era metodikoan egiten bada. Horrela jokatuz gero, ebazpide eraginkorragoa lor daiteke eta, era berean, azkeneko programa iteratiboaren zuzentasuna justifikatuta geratzen da.

Gainera, transformazio-eredu honen bidez sakonki azter daiteke *errekurtsibilitatearen eta iterazioaren arteko erlazioa*.

Gai honetan, *Burstall-en metodoa*<sup>1</sup> landuko dugu. Metodo honek funtzioren errekurtsiboko iteratiboa bihurtzeko balio du.

Honako hauek dira metodoaren oinarriak:

1. Abiapuntutzat, algoritmo errekurtsiboa edo zehaztapen induktiboa hartzen da.
2. Formulazio induktibo horri erreparatuta, soluzioaren *errekurrentzia-erlazioa* ateratzen da.

<sup>1</sup>A Transformation System for Developing Recursive Programs. R.M Burstall and J. Darlington. In *Journal of the ACM*, Volume 24, Issue 1, pages 44-67. January 1977.

3. Errekurrentzia-erlazio hori izango da, hain zuzen ere, oraindik sortu gabe da- goen iterazioaren *inbariantea*. Iterazioaren borne-adierazpena ere pauso honetan asmatzen da.
4. Inbariante hori oinarri hartuta, *iterazioa* edo algoritmo iteratiboa formalki era- tortzen da, honako urratsetan: *hasieraketa*, *bukaerako tratamendua* (*emaitza*) eta *iterazioaren gorputza*.

Iterazioaren gorputza eratortzeko, definizio induktiboan eta inbariantean oinarrituta, *destolesketa/tolesketa* teknika erabiltzen da.

## 6.2 Metodoaren deskribapena

Esan bezala, Burstall-en metodo orokorra funtziobaten definizio induktiboarekin has- ten da

$$f : T_1 \times \dots \times T_n \rightarrow T'$$

non  $\bar{T} = T_1 \dots T_n$  diren funtzioko parametro formalen motak, eta  $T'$  den emaitzaren mota.  $\bar{x} = x_1 \dots x_n$  deituko diegu parametro formalei ( $x_1 \in T_1, \dots, x_n \in T_n$ ). Definizio induktibo horretatik errekurrentzia-erlazio bat ateratzen da, eta erlazio hori inbariantetzat hartzen da. Honako hau izango da inbariantearen forma:

$$INB \equiv f(\bar{x}) = F(f(\bar{y}), \bar{z})$$

non  $\bar{y}$  eta  $\bar{z}$  aldagai berriak diren, iterazioan erabiliko direnak.  $F$  funtzioren bidez adierazten dugu  $f(\bar{y})$  eta  $\bar{z}$ -ren arteko erlazioa, hasierako funtzioren definizio indukti- botik asmatu behar duguna. Orokorean, errekurrentzia erlazioak antz handia hartzen du kasu induktiboeak bueltatzen duten emaitzarekin.

Borne-adierazpena  $\bar{y}$  aldagaien gainean definituko da, eta iterazioak bukatzeko falta dituen pausoen kopurua mugatuko du.

Inbariante horretatik eratorriko den funtzioboa honako forma hartzen du:

```
function f_it(x : T) return T' is
 y : T;
 z : ?;
begin
 -- Hasieraketa
 while not bukbal(y) loop INB ≡ { f(x) = F(f(y), z) }
 -- Gorputza
 (y, z) := (y', z');
 end loop;
 -- Emaitza
end f_it;
```

f\_it funtzioboa  $f$  funtziobaren goi-burukoa berdina da: parametro berdinak hartzen ditu ( $\bar{x}$ ) eta emaitza ere mota berekoa da. Hori bai, f\_it funtziok beste aldagai laguntzaile batzuk ere erabiltzen ditu:  $\bar{y}$  eta  $\bar{z}$ .  $\bar{x}$  parametroek konputazio errekurtsiboan hartuko lituzketen balioak gordetzeko erabiltzen dira  $\bar{y}$  aldagaiak algoritmo iteratiboa, eta horregatik dira mota berekoak.  $\bar{z}$  aldagaiak informazio osagarria gordetzeko erabiltzen dira.

Programaren lehenengo zatia hasieraketa da, eta bertan  $\bar{y}$  eta  $\bar{z}$  aldagaiak hasieratu behar dira inbariantea iteraziora sartzerako bete dadin.

Programaren bigarren zatia iterazioa bera da, non bukaera-baldintza (*bukbal* funtzioa) eta gorputza definitu behar diren. Bukaera-baldintza definitzeko, hasierako algoritmo errekurtsiboko kasu nabariekin parekatzen dira iterazioko bukaera-kasuak. Oinarria simplea da: iterazioa bukatuko da errekurtsioa bukatuko litzatekeen egoera berdinetan. Hori bai, kontuan hartu behar da errekurtsioaren bukaera  $\bar{x}$  parametroen arabera formulatzen zela, eta iterazioan, berriz,  $\bar{y}$  aldagaien arabera formulatu beharko da. Hor-taz, *bukbal* funtzioa kasu nabari guztiak baldintzen disjuntzioa izango da. Iterazioaren gorputzari dagokionez, irizpide nagusia honako hau da: iterazioko pauso bakoitzak dei errekurtsibo bat *simulatu* behar du. Nolabait, algoritmo errekurtsiboaren exekuzioan dei batetik bestera gertatzen ziren aldaketak izango dira  $\bar{y}$  eta  $\bar{z}$  aldagaietan jasoko direnak. Aldagai horiei, beraz, balio berriak esleitzen zaizkie iterazio-pauso bakoitzean. Metodoaren aplikazioan, balio berri horiek zaharretatik bereizteko horrela izendatuko dira:  $\bar{y}'$  eta  $\bar{z}'$ . Balio horiek destolesketa/tolesketa teknikaren bidez lortzen dira inbariantea oinarritzat hartuta eta kasu induktiboa arabera. Behin balio berriak zein izango diren erabakita, erraz asma daiteke gorputzaren kodea: azken batean,  $\bar{y}$  eta  $\bar{z}$  aldagaiei balio berriak esleitu beharko zaizkie. Asignazio horiek, formalki, aldi berean gertatzen dira eta  $(\bar{y}, \bar{z}) := (\bar{y}', \bar{z}')$  notazioarekin adieraz daiteke aldi berekoak direla. Baina, ADAn asignazioak banan-banan egin beharra dagoenez, kontu handia eduki behar da asignazioen hurrenkerarekin, eta batzuetan aldagai laguntzaileak ere erabili behar dira. Gainera, kasu induktiboa bat baino gehiago dagoenean, kasu horien araberako esleipen baldintzatuak egin behar izaten dira, eta horregatik, *if* agindua erabiltzen da.

Programaren azken zatian bukaerako tratamendua egiten da, hau da, azken emaitza lortzea. Emaitza zein den kalkulatzeko inbariantearen ebaluazioa egiten da  $\bar{y}$  eta  $\bar{z}$  aldagaien balioekin (iteraziotik ateratakoan izango dituzten balioekin). Honakoan ere, kasu nabari bat baino gehiago badago, kasu horiek denak ondo tratatzeko *if* agindu bat erabiltzen da.



### 6.3 Metodoaren aplikazio-adibideak

Lehenengo adibidea zenbakien oinarri-aldaketa da. Izan bedi  $oinald(x, b)$  funtzioa,  $x$  zenbaki arruntaren  $b$  oinarriko errepresentazioa itzultzen duena:

$$oinald: \text{Integer} \times \text{Integer} \rightarrow \text{sekuentzia(Integer)}$$

$$\text{Aurre: } 1 < b < 10 \wedge x \geq 0$$

$$oinald(x, b) = \begin{cases} \langle x \rangle & \text{baldin } x < b \\ oinald(x/b, b) @ \langle x \bmod b \rangle & \text{bestela} \end{cases}$$

Definizio errekurtsibo hori programa iteratibo bihurtuko dugu.

1. Hasteko, errekurrentzia-erlaziona atera behar dugu. Helburu horretara iristeko hurbilpen egokia da destolesketa/tolesketa teknika erabiltzea adibide zehatz bat-en gainean. Adibidez,  $oinald(123, 4)$  adibidea hartuko dugu, eta espresio horren kalkulua egingo dugu definizio induktiboak adierazitakoaren arabera. Kalkulu horretan, destolesketa-urrats bakoitzean funtziaren definizioa aplikatzen da eta, berriz, tolesketa-urrats bakoitzen, lortzen dugun adierazpena simplifikatu egiten da.  
 $oinald(123, 4)$  destolestuta  $oinald(30, 4) @ \langle 3 \rangle$  lortzen dugu ( $oinald$ -en kasu induktiboa), eta adierazpen hau ezin da simplifikatu. Ondoren, beste destolesketa-urrats bat egingo dugu, orain  $oinald(30, 4)$ -ren gainean, eta  $oinald(7, 4) @ \langle 2 \rangle$  ateratzen da (kasu induktiboa). Adierazpen osoa  $(oinald(7, 4) @ \langle 2 \rangle) @ \langle 3 \rangle$  da eta,

sekuentzien kateaketa elkarkorra denez, simplifika daiteke (tolesketa-urratsa):  $oinald(7, 4)@ \langle 2, 3 \rangle$ . Berriro beste destolesketa-urrats bat egingo da  $oinald(7, 4)$ -ren gainean, eta, horrela, bukatu arte. Laburbilduz, destolesketa/tolesketa teknika erabiliz gelditzen den transformazioa horrelakoa da:

$$\begin{aligned}
& \underbrace{oinald(123, 4)}_{destolesu/tolestu} \\
& = \underbrace{oinald(30, 4) @ \langle 3 \rangle}_{destolesu} \\
& = \underbrace{(oinald(7, 4) @ \langle 2 \rangle) @ \langle 3 \rangle}_{tolestu} \\
& = \underbrace{oinald(7, 4) @ \langle 2, 3 \rangle}_{destolesu} \\
& = \underbrace{(oinald(1, 4) @ \langle 3 \rangle) @ \langle 2, 3 \rangle}_{tolestu} \\
& = \underbrace{oinald(1, 4) @ \langle 3, 2, 3 \rangle}_{destolesu} \\
& = \underbrace{\langle 1 \rangle @ \langle 3, 2, 3 \rangle}_{tolestu} \\
& = \langle 1, 3, 2, 3 \rangle
\end{aligned}$$

Orain, transformazio hori aztertzen da errekurrentzia ateratzeko. Horretarako, tolesketa-urrats guztien ondoren gelditzen den adierazpena kontuan hartzen dugu:

$$\begin{aligned}
& oinald(30, 4) @ \langle 3 \rangle \\
& oinald(7, 4) @ \langle 2, 3 \rangle \\
& oinald(1, 4) @ \langle 3, 2, 3 \rangle \\
& \langle 1, 3, 2, 3 \rangle
\end{aligned}$$

Azkena izan ezik, oso erraz ikus daiteke gelditzen den adierazpenaren forma orokorra horrelakoa dela

$$oinald(x, b) = oinald(y, b) @ S$$

non  $x$ ,  $b$  eta  $y$  zenbaki arruntak diren, eta  $S$  zenbaki arrunten sekuentzia bat. Hau da, hasierako espresioa beti da zenbaki baten oinarri-aldaketaren emaitza zenbaki osoko sekuentzia batekin kateatuta. Adierazpen hori da atera dugun errekurrentzia-erlazioa eta hori izango da, hain zuzen, inbarianetzat hartuko duguna. Garbi ikusten da inbariantea eta kasu induktiboaren emaitza oso antzekoak direla.

Borne-adierazpena ere asma dezakegu egin dugun transformazioan oinarrituta. Destolesketa/tolesketa prozesuan ikusi dugu  $y$  aldagaia gero eta txikiagoa egiten dela, harik eta  $b$  baino txikiagoa izan eta prozesua bukatu den arte. Borne-adierazpena, hortaz, honako hau izan daiteke:  $E \equiv y$

2. Ondoren, eta inbariantea abiapuntutzat hartuta, programa iteratiboaren atal guztiak eratortzen dira:

- Hasieraketa. Ataza honek ziurtatu behar du inbariantea betetzen dela lehenengo iterazioa egikaritu baino lehenago. Horretarako, pentsatu behar dugu zein balio hartu behar duten  $y$  eta  $S$  aldagaietan:

$$oinald(x, b) = oinald(y, b)@S$$

inbariantea betetzeko. Kasu honetan, soluziorik errazena honako hau da:

$$\begin{aligned} y &:= x; \\ S &:= \langle \rangle; \end{aligned}$$

- Bukaera-baldintza. Hasierako definizio induktiboan kasu nabari bakar bat dago,  $x < b$  baldintzaren bidez adierazten dena. Iterazioan, hortaz, bukaera-baldintza simplea izango da, eta horrela formulatuko da:  $bukbal(y) \equiv y < b$ .
- Emaitzia. Bukaera-baldintza bakarra dagoenez, emaitza ere modu bakunean adieraz dezakegu (*if* aginduaren beharrik gabe). Inbarianean  $oinald(y, b)$  adierazpena bere balioarekin ( $\langle y \rangle$ ) ordeztuta lortuko dugu emaitza:

$$y < b \rightarrow oinald(x, b) = oinald(y, b)@S = \langle y \rangle @ S = y \bullet S$$

Hau da, emaitza  $y \bullet S$  da. ADAn adierazita:

return  $y \bullet S;$

- Iterazioaren gorputza. Inbariantea abiapuntu hartuta, destolesketa / tolesketa teknika erabiliko dugu kasu induktibo bakarrarekin:

$$\begin{aligned} oinald(x, b) &= \underbrace{oinald(y, b)}_{destolesku} @ S \\ &= \underbrace{(oinald(y/b, b) @ \langle y \bmod b \rangle)}_{tolesku} @ S \\ &= oinald(\underbrace{y/b, b}_{y', b}) @ \underbrace{\langle y \bmod b \rangle @ S}_{S'} \\ &= oinald(y', b) @ S' \end{aligned}$$

Hau da, iterazioen gorputza honako aldibereko asignazioa da:

$$(S, y) := (\langle y \bmod b \rangle @ S, y/b);$$

Aldibereko asignazio anizkoitz honen asignazioak ezin daitezke edozein ordenatan egin.  $S$  sekuentzia eguneratzeko erabiltzen den  $y$  aldagaiaren balioa zaharra da,  $y$  eguneratu aurrekoan. Beraz, beharrezko da  $y$  eguneratu aurriko  $S$  eguneratzea:

$$\begin{aligned} S &:= \langle y \bmod b \rangle @ S; \\ y &:= y/b; \end{aligned}$$

Horrenbestez, metodoa aplikatuta lortu dugun funtziotako itzulpena honako hau da:

```
function oinald_it(x, b : Integer) return sekuentzia(Integer) is
 y : Integer;
 S : sekuentzia(Integer);
begin { 1 < b < 10 ∧ x ≥ 0 }
```

Ikusiko dugun bigarren adibidea sekuentzien nahasketa egitean datza. Adibide honetan deskribatzen da zer gertatzen den kasu induktiboetan erabiltzen diren eragiketak ez-elkarkorrik direnean.

Izan bedi  $nahastu$  funtzioa, non  $nahastu(S, R)$  adierazpenak  $S$  eta  $R$  sekuentzien fusioa itzultzen duen:

$$nahastu: \text{sekuentzia}(T) \times \text{sekuentzia}(T) \rightarrow \text{sekuentzia}(T)$$

$$(1) \ nahastu(\langle \rangle, R) = R$$

$$(2) \ nahastu(S, \langle \rangle) = S$$

$$(3) \ nahastu(x \bullet S, y \bullet R) = \begin{cases} x \bullet nahastu(S, y \bullet R) & \text{baldin } x \leq y \\ y \bullet nahastu(x \bullet S, R) & \text{bestela} \end{cases}$$

Funtzio honek bi kasu nabari eta bi kasu induktibo ditu. Aplika dezagun Burstall-en metodoa, definizio induktibo horretatik funtzio iteratibo bat lortzeko.

1. Aurreko adibidean egin dugun bezalaxe, destolesketa/tolesketa teknika erabiliko dugu datu zehatz batzuen gainean erreurrentzia erlazioa ateratzeko. Baino, adibide honetan, datuak aukeratzean kasu induktibo guztiak gerta daitezten bi-latuko dugu. Esate baterako,  $nahastu(\langle 1, 4, 6 \rangle, \langle 2, 3 \rangle)$  adibide on bat da, 1 (lehenengo sekuentziako lehenengo osagaia) 2 (bigarren sekuentziako lehenengo osagaia) baino txikiagoa delako (lehenengo kasu induktiboa), eta 4 (lehenengo sekuentziako bigarren osagaia) 2 baino handiagoa delako. Alegia, definizio induktiboaren araberako kalkuluan, bi kasu induktiboak gertatuko dira, eta horrek orokorta-suna ematen dio egindako transformazioari.

Hala bada,  $nahastu(\langle 1, 4, 6 \rangle, \langle 2, 3 \rangle)$  destolestu ondoren  $1 \bullet nahastu(\langle 4, 6 \rangle, \langle 2, 3 \rangle)$  lortzen dugu (funtzioaren lehenengo kasu induktiboa). Simplifikatu ezin denez,  $nahastu(\langle 4, 6 \rangle, \langle 2, 3 \rangle)$  destolesten dugu, eta  $2 \bullet nahastu(\langle 4, 6 \rangle, \langle 3 \rangle)$  lortzen dugu. Adierazpen osoa  $1 \bullet (2 \bullet nahastu(\langle 4, 6 \rangle, \langle 3 \rangle))$  da, eta adierazpen hau ezin da simplifikatu • eragiketa erabiliz. Kontua da • ez-elkarkorra dela eta, horregatik, ezin ditugula 1, 2 eta  $nahastu(\langle 4, 6 \rangle, \langle 3 \rangle)$  adierazpenak beste era batean •-ren bidez erlazionatu. Kasu hauetan, funtzio ez-elkarkorra beste funtzio batekin ordezkatzen da, elkarkorra dena. Sekuentziak osatzeko elkarkorra den funtzio bat kateaketa da, elkarkorra dena. Hontzaz, • funtzioa kateaketarekin (@) ordezkatuta, destolesketa/tolesketa prozesu osoa horrela gelditzen da:

$$\begin{aligned} &\underbrace{nahastu(\langle 1, 4, 6 \rangle, \langle 2, 3 \rangle)}_{\text{destolestu}} \\ &= \underbrace{1 \bullet nahastu(\langle 4, 6 \rangle, \langle 2, 3 \rangle)}_{\text{tolestu}} \end{aligned}$$

- 4.9. Sekuentzia bat emanda, osagai berdinez osatutako azpisekuentzia bakoitza azpisekuentziaren osagai batekin ordeztu. Adibidez,  $\langle 1, \underline{5}, 5, 5, 7, 9, 5, \underline{6}, 6, 5 \rangle$  emanda, ondokoa izango da emaitza:  $\langle 1, \underline{5}, 7, 9, 5, \underline{6}, 5 \rangle$
- 4.10. Bi sekuentzia ordenatuen nahasketa ordenatua eman.
- 4.11. Zenbakি oso positiboz osatutako  $s$  sekuentzia emanda,  $s$  sekuentziaren balio bikoiez osatutako pila eman.
- 4.12. *aurrizki* funtziak,  $S$  sekuentzia eta  $n$  zenbakি arrunta emanda,  $S$ -ko  $n$  hasierako elementuez osatutako azpi-sekuentzi itzultzen du.  $S$  sekuentziak  $n$  elementu ez badauzka, orduan  $S$  sekuentzia itzuli.

Adibidez:

- $\text{aurrizki}(\langle 4, 4, 2, 1, 1, 8 \rangle, 3) = \langle 4, 4, 2 \rangle$
- $\text{aurrizki}(\langle 4, 4, 2, 1, 1, 8 \rangle, 9) = \langle 4, 4, 2, 1, 1, 8 \rangle$

Kendu:  $\text{pile}(T) \times T \rightarrow \text{pile}(T)$

$$P = \begin{array}{|c|} \hline 4 \\ \hline 5 \\ \hline 6 \\ \hline 3 \\ \hline \end{array} \quad x = 6 \quad \text{Kendu}(P, x) = \begin{array}{|c|} \hline 6 \\ \hline 3 \\ \hline \end{array}$$

Bahetu:  $\text{pile}(T) \times T \times \text{pile}(T) \rightarrow \text{pile}(T)$

$$P = \begin{array}{|c|} \hline 4 \\ \hline 5 \\ \hline 6 \\ \hline 3 \\ \hline 5 \\ \hline 3 \\ \hline \end{array} \quad x = 4 \quad q = \begin{array}{|c|} \hline \\ \hline ? \\ \hline q \\ \hline 8 \\ \hline \end{array} \quad \text{Bahetu}(P, x, q) = \begin{array}{|c|} \hline 3 \\ \hline 3 \\ \hline 4 \\ \hline 2 \\ \hline 9 \\ \hline 8 \\ \hline \end{array}$$

## 4. Gaia: Espezifikazio ekuazionala

### 6. Ariketa-orria (B)

3. *aurrizki* tuntziorako ematen diren espezifikazio ekuazionalak aztertu itzazu eta arazoranatu ebazen bakotza zuzena den ala ez:

*aurrizki* funtzioak,  $S$  sekuentzia eta  $n$  zenbaki arrunta emanda,  $S$ -ko  $n$  hasierako elementuez osatutako azpi-sekuentzi itzultzen du.  $S$  sekuentziak  $n$  elementu ez badauzka, orduan  $S$  sekuentzia itzultzen du.

Adibidez:

- $\text{aurrizki}(\langle 4, 4, 2, 1, 1, 8 \rangle, 3) = \langle 4, 4, 2 \rangle$
- $\text{aurrizki}(\langle 4, 4, 2, 1, 1, 8 \rangle, 9) = \langle 4, 4, 2, 1, 1, 8 \rangle$

Mota:  $\text{sekuentzia}(T)$

Lag: (ez dago)

Eragiketa:

$\text{aurrizki}: \text{sekuentzia}(T) \times \text{nat} \rightarrow \text{sekuentzia}(T)$

3.1. Ekuazioak:

- (1)  $\text{aurrizki}(S, 0) = \langle \rangle$
- (2)  $\text{aurrizki}(x \bullet S, n + 1) = x \bullet \text{aurrizki}(S, n)$

3.2. Ekuazioak:

- (1)  $\text{aurrizki}(S, 0) = \langle \rangle$
- (2)  $\text{aurrizki}(\langle \rangle, n + 1) = \langle \rangle$
- (3)  $\text{aurrizki}(x \bullet S, n + 1) = x \bullet \text{aurrizki}(S, n)$

3.3. Ekuazioak:

- (1)  $\text{aurrizki}(\langle \rangle, 0) = \langle \rangle$
- (2)  $\text{aurrizki}(\langle \rangle, n) = \langle \rangle$
- (3)  $\text{aurrizki}(x \bullet S, n) = x \bullet \text{aurrizki}(S, n - 1)$

4 *sekuentzia* DMA aberasteko asmoz, ondoren deskribatzenten diren funtzioen espezifikazio ekuazionala idatz ezazu (urretik definitutako eragiketak erabil daitezke):

~~4.1.~~ Elementu bat arbola bitar batean zenbat aldiz agertzen den erabaki.

4.2. Zenbaki arruntez osatutako sekuentzia baten osagaiak inkrementatu bat balioarekin.

4.3. Sekuentzia batean osagai errepikatuak dauden ala ez erantzun.

4.4. Sekuentzia baten elementu errepikatuak ezabatu ezkerrerago dagoen osagaia utziz.

4.5. Sekuentzia baten elementu errepikatuak ezabatu eskuinerago dagoen osagaia utziz.

4.6. Sekuentzia batean ondoz-ondoko bi osagai berdin, gutxienez behin, agertzen diren erantzun.

4.7. Sekuentzia bat beste baten hasierako *aurrizkia* den ala ez erantzun.

4.8. *bikoiztu* funtzioak banan-banan *bikoitzen* ditu sekuentzia bateko elementu guztia.

Adibidez,  $\text{bikoiztu}(\langle a, d, k, f \rangle)$  funtzioaren emaitza  $\langle a, a, d, d, k, k, f, f \rangle$  da.