5

Programa errekurtsiboen transformazioa: Burstall-en metodoa

5.1 Burstall-en metodoa, programa errekurtsiboak iteratibo bihurtzeko

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

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

Baina askotan, batez ere programa iteratiboekin konparatuz, 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 abiapuntu 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 *errekurtsibitatearen eta iterazioaren arteko erlazioa.*

Gai honetan, *Burstall-en metodoa*[1] landuko dugu. Metodo honek funtzio errekurtsiboak iteratibo 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.

[1] *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 dagoen iterazioaren *inbariantea*. Iterazioaren borne-adierazpena ere pauso honetan asmatzen da.

4. Inbariante hori oinarri hartuta, *iterazioa* edo algoritmo iteratiboa formalki eratortzen da, honako urratsetan: *hasieraketa*, *bukaerako tratamendua* (*emaitza*) eta *iterazioaren gorputza*.

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

5.2 Metodoaren deskribapena

Esan bezala, Burstall-en metodo orokorra funtzio baten definizio induktiboarekin hasten da

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

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

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

non $\overline{y}$ eta $\overline{z}$ aldagai berriak diren, iterazioan erabiliko direnak. F funtzioaren bidez adierazten dugu $f(\overline{y})$ eta $\overline{z}$-ren arteko erlazioa, hasierako funtzioaren definizio induktibotik asmatu behar duguna. Orokorrean, errekurrentzia erlazioak antz handia hartzen du kasu induktiboek bueltatzen duten emaitzarekin.

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

Inbariante horretatik eratorriko den funtzio iteratiboak 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               E ≡ g(y)
      (y,z) := (y',z');
   end loop;
   - - Emaitza
end f_it;
```

f_it funtzio iteratiboaren goi-burukoa f funtzio errekurtsiboaren berdina da: parametro berdinak hartzen ditu ($\overline{x}$) eta emaitza ere mota berekoa da. Hori bai, *f_it* funtzioak beste aldagai laguntzaile batzuk ere erabiltzen ditu: $\overline{y}$ eta $\overline{z}$. $\overline{x}$ parametroek konputazio errekurtsiboan hartuko lituzketen balioak gordetzeko erabiltzen dira $\overline{y}$ aldagaiak algoritmo iteratiboan, eta horregatik dira mota berekoak. $\overline{z}$ aldagaiak informazio osagarria gordetzeko erabiltzen dira.

Programaren lehenengo zatia hasieraketa da, eta bertan $\overline{y}$ eta $\overline{z}$ 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 sinplea da: iterazioa bukatuko da errekurtsioa bukatuko litzatekeen egoera berdinetan. Hori bai, kontuan hartu behar da errekurtsioaren bukaera $\overline{x}$ parametroen arabera formulatzen zela, eta iterazioan, berriz, $\overline{y}$ aldagaien arabera formulatu beharko da. Hortaz, *bukbal* funtzioa kasu nabari guztien 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 $\overline{y}$ eta $\overline{z}$ aldagaietan jasoko direnak. Aldagai horiei, beraz, balio berriak esleitzen zaizkie iterazio-pauso bakoitzean. Metodoaren aplikazioan, balio berri horiek zaharretatik bereizteko horrela izendatuko dira: $\overline{y}'$ eta $\overline{z}'$. Balio horiek destolesketa/tolesketa teknikaren bidez lortzen dira inbariantea oinarritzat hartuta eta kasu induktiboen arabera. Behin balio berriak zein izango diren erabakita, erraz asma daiteke gorputzaren kodea: azken batean, $\overline{y}$ eta $\overline{z}$ aldagaiei balio berriak esleitu beharko zaizkie. Asignazio horiek, formalki, aldi berean gertatzen dira eta $(\overline{y}, \overline{z}) := (\overline{y}', \overline{z}')$ 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 induktibo 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 $\overline{y}$ eta $\overline{z}$ 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.

5.3 Metodoaren aplikazio-adibideak

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

$$
\begin{array}{l}
oinald\text{: } Integer \times Integer \rightarrow sekuentzia(Integer) \\
\underline{\text{Aurre}}\text{: } 1 < b < 10 \ \wedge \ x \geq 0 \\
oinald(x,b) = \left\{ \begin{array}{ll} \langle x \rangle & baldin\ x < b \\ oinald(x/b, b) @ \langle x \ mod \ b \rangle & bestela \end{array} \right.
\end{array}
$$

Definizio errekurtsibo hori programa iteratibo bihurtuko dugu.

1. Hasteko, errekurrentzia-erlazioa atera behar dugu. Helburu horretara iristeko hurbilpen egokia da destolesketa/tolesketa teknika erabiltzea adibide zehatz baten gainean. Adibidez, $oinald(123, 4)$ adibidea hartuko dugu, eta espresio horren kalkulua egingo dugu definizio induktiboak adierazitakoaren arabera. Kalkulu horretan, destolesketa-urrats bakoitzean funtzioaren definizioa aplikatzen da eta, berriz, tolesketa-urrats bakoitzen, lortzen dugun adierazpena sinplifikatu egiten da.

   $oinald(123, 4)$ destolestuta $oinald(30, 4) @ \langle 3 \rangle$ lortzen dugu ($oinald$-en kasu induktiboa), eta adierazpen hau ezin da sinplifikatu. 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, sinplifika 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:

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

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

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

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 osokoen sekuentzia batekin kateatuta. Adierazpen hori da atera dugun errekurrentzia-erlazioa eta hori izango da, hain zuzen, inbariantetzat 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 aldagaiek

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

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

     ```
     y := x;
     S := <>;
     ```

   - Bukaera-baldintza. Hasierako definizio induktiboan kasu nabari bakar bat dago, $x < b$ baldintzaren bidez adierazten dena. Iterazioan, hortaz, bukaera-baldintza sinplea izango da, eta horrela formulatuko da: $bukbal(y) \;\equiv\; y < b$.

   - Emaitza. Bukaera-baldintza bakarra dagoenez, emaitza ere modu bakunean adieraz dezakegu (*if* aginduaren beharrik gabe). Inbariantean $oinald(y, b)$ 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 • S;
     ```

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

     $$\begin{aligned}
     &oinald(x, b) \\
     &\quad = \underbrace{oinald(y, b)}_{destolestu} @S \\
     &\qquad = \underbrace{(oinald(y/b, b)@\langle y \bmod b \rangle)@S}_{tolestu} \\
     &\quad = oinald(\underbrace{y/b}_{y'}, b)@\underbrace{(\langle y \bmod b \rangle @S)}_{S'} \\
     &\quad = oinald(\; y', \; b)@
     \end{aligned}$$

     Hau da, iterazioen gorputza honako aldibereko asignazioa da:

     ```
     (S,y) := (< y mod b > @ 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 aurrekoa. Beraz, beharrezkoa da y eguneratu aurretik S eguneratzea:

     ```
     S := < y mod b > @ S;
     y := y/b;
     ```

Horrenbestez, metodoa aplikatuta lortu dugun funtzio iteratiboa honako hau da:

```
function oinald_it(x,b :  Integer) return sekuentzia(Integer) is
   y :  Integer;
   S : sekuentzia(Integer);
begin     { 1 < b < 10 ∧ x ≥ 0 }
   y := x;
   S := <>;
   while not y < b loop      INB ≡ { oinald(x,b) = oinald(y,b)@S }
                             E ≡ y
      S := < y mod b > @ S;
      y := y/b;
   end loop;
   return y • S;
end oinald_it;
{ oinald_it(x,b) = oinald(x,b) }
```

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

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

$$nahastu\colon\ sekuentzia(T) \times sekuentzia(T) \to sekuentzia(T)$$

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

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

1. Aurreko adibidean egin dugun bezalaxe, destolesketa/tolesketa teknika erabiliko dugu datu zehatz batzuen gainean errekurrentzia erlazioa ateratzeko. Baina, adibide honetan, datuak aukeratzean kasu induktibo guztiak gerta daitezen bilatuko 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 orokortasuna 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). Sinplifikatu 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 sinplifikatu $\bullet$ eragiketa erabiliz. Kontua da $\bullet$ ez-elkarkorra dela eta, horregatik, ezin ditugula 1, 2 eta $nahastu(\langle 4,6\rangle, \langle 3\rangle))$ adierazpenak beste era batean $\bullet$-ren bidez erlazionatu. Kasu hauetan, funtzio ez-elkarkorra beste funtzio batekin ordezkatzen da, elkarkorra dena. Sekuentziak osatzeko elkarkorra den funtzio bat kateaketa (@) da eta, horren bidez, aurreko adierazpena toles daiteke: $\langle 1,2\rangle @ nahastu(\langle 4,6\rangle, \langle 3\rangle)$. Hortaz, $\bullet$ funtzioa kateaketarekin (@) ordezkatuta, destolesketa/tolesketa prozesu osoa horrela gelditzen da:

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

Transformazio hori aztertu ondoren ateratzen den errekurrentzia-erlazioa honako hau da

$$nahastu(S,R) \;=\; U @ nahastu(V,W)$$

non S, R, U, V eta W zenbaki osokoen sekuentziak diren. U, V eta W aldagai berriak dira. Ikus daitekeenez, kasu induktiboetan erabiltzen diren adierazpenetatik atera da errekurrentzia-erlazioa, $\bullet$ funtzioa kateaketarekin ordezkatuz. Errekurrentzia-erlazio hori funtzio iteratiboaren inbariantea izango da.

Borne-adierazpenari dagokionez, esan dezakegu bukaerarako falta dena V eta W sekuentzien luzeraren araberakoa dela. Aukera bat baino gehiago badaude ere, borne-adierazpen egokia da honako hau: $E \equiv luz(V) + luz(W)$, non $luz(V)$ eta $luz(W)$ V eta W sekuentzien luzerak diren hurrenez hurren.

2. Ondoren, eta inbariantea ardatz hartuta, programa iteratiboaren zati guztiak eratorriko ditugu:

   - Hasieraketa. *while*-ra lehenengo aldiz iristean inbariantea bete dadin, honako asignazio hauek egin daitezke hasieran:

     ```
     U := <>;
     V := S;
     W := R;
     ```

   - Bukaera-baldintza. Bi kasu nabari daudenez, horiek ondo jasotzen dituen bukaera-baldintza disjuntzio baten bidez adieraziko dugu, hasierako parametroen ordez iterazioko aldagaiak jarriz:

     $$bukbal(V,W) \;\equiv\; hutsa\_da?(V) \;\vee\; hutsa\_da?(W)$$

     Iteraziotik aterako gara V edo W sekuentzia hutsa denean.

- Emaitza. Bukaera-baldintza bi kasuren disjuntzioa denez, kasu bakoitzaren emaitza era banatuan kalkulatuko dugu. Inbariantean $nahastu(V, W)$ adierazpenak duen balioa jarriko dugu kasu bakoitzean:

$$hutsa\_da?(V) \rightarrow nahastu(S, R) = U@nahastu(V, W) = U@W$$
$$hutsa\_da?(W) \rightarrow nahastu(S, R) = U@nahastu(V, W) = U@V$$

  Hau da, emaitza $U@W$ edo $U@V$ izango da, bukaerako kasuaren arabera:

```
if hutsa_da?(V) then
   return U @ W;
else
   return U @ V;
end if;
```

- Iterazioaren gorputza. Inbariantea abiapuntu hartuta, destolesketa / tolesketa teknika erabiliko dugu kasu induktibo bakoitzeko.
  Lehenengo kasu induktiboa: $lehena(V) \leq lehena(W)$

$$\begin{aligned} &nahastu(S, R) \\ &\quad = U@\underbrace{nahastu(V, W)}_{destolestu} \\ &\qquad = \underbrace{U@(lehena(V)@nahastu(hondarra(V), W))}_{tolestu} \\ &= \underbrace{(U@\langle lehena(V)\rangle)}_{U'}@nahastu(\underbrace{hondarra(V)}_{V'}, \underbrace{W}_{W'}) \\ &= \qquad U' \qquad @nahastu(\quad V', \quad W') \end{aligned}$$

  Hau da, lehenengo kasu induktiboan iterazioaren gorputza honako aldibereko asignazioa da:

```
(U,V,W) := (U @ < lehena(V) >,hondarra(V),W);
```

  Aldibereko asignazio hau era bakar batean inplementa daiteke

```
U := U @ <lehena(V)>;
V := hondarra(V);
```

  Izan ere, lehenago $V := hondarra(V)$; jarriko bagenu, orduan V sekuentzian agertzen den lehenengo osagaia galdu egingo genuke.
  Bigarren kasu induktiboa: $lehena(V) > lehena(W)$

$$\begin{aligned} &nahastu(S, R) \\ &\quad = U@\underbrace{nahastu(V, W)}_{destolestu} \\ &\qquad = \underbrace{U@(lehena(W)@nahastu(V, hondarra(W)))}_{tolestu} \\ &= \underbrace{(U@\langle lehena(W)\rangle)}_{U'}@nahastu(\underbrace{V}_{V'}, \underbrace{hondarra(W)}_{W'}) \\ &= \qquad U' \qquad @nahastu(\ V', \quad W') \end{aligned}$$

  Hortaz, aldibereko asignazioa honako hau da:

```
(U,V,W) := (U @ <lehena(W)>,V,hondarra(W));
```

eta horrela inplementa daiteke:

```
U := U @ <lehena(W)>;
W := hondarra(W);
```

Bi kasuak bilduta, iterazioaren gorputza horrela gelditzen da:

```
if lehena(V) <= lehena(W) then
   U := U @ <lehena(V)>;
   V := hondarra(V);
else
   U := U @ <lehena(W)>;
   W := hondarra(W);
end if;
```

Lortu dugun funtzio iteratiboa honako hau da:

```
function nahastu_it(S,R : sekuentzia(T)) return sekuentzia(T) is
   U,V,W : sekuentzia(T);
begin
   U := <>;
   V := S;
   W := R;
                        INB ≡ { nahastu(S,R) = U@nahastu(V,W) }
                        E ≡ luz(V)+luz(W)
   while not ( hutsa_da?(V) or hutsa_da?(W) ) loop
      if lehena(V) <= lehena(W) then
         U := U @ <lehena(V)>;
         V := hondarra(V);
      else
         U := U @ <lehena(W)>;
         W := hondarra(W);
      end if;
   end loop;
   if hutsa_da?(V) then
      return U @ W;
   else
      return U @ V;
   end if;
end nahastu_it;
{ nahastu_it(S,R) = nahastu(S,R) }
```

Hirugarren adibidea aztertuko dugu orain. Honako honetan errekurtsibitate anizkoitzeko definizio induktibo bat transformatu, eta iteratibo bihurtuko dugu. Horretarako, destolestean dei bat baino gehiago eduki behar dira kontuan. Gainera, aztertu behar da zein dei-konbinazio destolestu behar den, gero tolestu ahal izateko.

Izan bedi fib funtzioa, non $fib(n)$ adierazpenak *Fibonacci* segidaren n-garren zenbakia bueltatzen duen:

fib: $Integer \rightarrow Integer$

Aurre: $n \geq 0$

$$fib(n) = \begin{cases} n & baldin\ n \leq 1 \\ fib(n-1) + fib(n-2) & bestela \end{cases}$$

Funtzio honek kasu nabari bakar bat eta kasu induktibo bakar bat ditu, baina kasu induktiboan bi dei errekurtsibo erabiltzen dira.

1. Aurreko adibideetan bezalaxe, destolesketa/tolesketa teknika erabiliko dugu errekurrentzi erlazioa ateratzeko, baina, kasu honetan, bi dira landu beharreko dei errekurtsiboak. Transformazioan estrategia hau erabiliko dugu: argumentu handiena duen deia destolestu.

$$\begin{aligned}
&\underbrace{fib(5))}_{destolestu} \\
&\quad = \underbrace{fib(4)}_{destolestu} + fib(3) \\
&\qquad = \underbrace{(fib(3) + fib(2)) + fib(3)}_{tolestu} \\
&\quad = 2 \times \underbrace{fib(3)}_{destolestu} + fib(2) \\
&\qquad = \underbrace{2 \times (fib(2) + fib(1)) + fib(2)}_{tolestu} \\
&\quad = 3 \times \underbrace{fib(2)}_{destolestu} + 2 \times fib(1) \\
&\qquad = \underbrace{3 \times (fib(1) + fib(0)) + 2 \times fib(1)}_{tolestu} \\
&\quad = 5 \times \underbrace{fib(1)}_{destolestu} + 3 \times \underbrace{fib(0)}_{destolestu} \\
&\qquad = 5 \times 1 + 3 \times 0 \\
&\quad = 5
\end{aligned}$$

Transformazio horri erreparatuta, errekurrentzia-erlazio bat baino gehiago asma daiteke:

$$\begin{aligned}
&(1) \quad fib(n) = x \times fib(u) + y \times fib(v) \\
&(2) \quad fib(n) = x \times fib(u) + y \times fib(u-1) \\
&(3) \quad fib(n) = x \times fib(v+1) + y \times fib(v)
\end{aligned}$$

Azken errekurrentzia-erlazioa hartuko dugu inbariantetzat:

$$INB \equiv fib(n) = x \times fib(v+1) + y \times fib(v)$$

Borne-adierazpena: $E \equiv v$

2. Inbariantea ardatz hartuta eratorriko ditugu programa iteratiboaren atal guztiak.

   - Hasieraketa. Inbariantea hasieran bete dadin, honako asignazio hauek erabil daitezke:

     ```
     x := 0;
     y := 1;
     v := n;
     ```

   - Bukaera-baldintza. Kasu nabari bakar bat dagoenez, bukaera-baldintza sinplea da:

     $$bukbal(v) \equiv v = 0$$

   - Emaitza. Kasu nabaria $v = 0$ da. Inbariantean $fib(v+1)$ eta $fib(v)$ adierazpenak beren une horretako balioarekin ordezkatuko ditugu:

     $$v = 0 \rightarrow fib(n) = x \times fib(v+1) + y \times fib(v) = x \times 1 + y \times 0 = x$$

     Hau da, emaitza x da:

     ```
     return x;
     ```

   - Iterazioaren gorputza. Inbariantea oinarritzat hartuta, destolesketa / tolesketa teknika erabiliko dugu:

     $$\begin{aligned} & fib(n) \\ & \quad = x \times \underbrace{fib(v+1)}_{destolestu} + y \times fib(v) \\ & \qquad = \underbrace{x \times (fib(v) + fib(v-1)) + y \times fib(v)}_{tolestu} \\ & = \underbrace{(x+y)}_{x'} \times fib(\underbrace{\quad v \quad}_{v'+1}) + \underbrace{x}_{y'} \times fib(\underbrace{v-1}_{v'}) \\ & = \quad x' \quad \times fib(\ v'+1\ ) + \ y' \ \times fib(\quad v' \quad) \end{aligned}$$

     Hortaz, iterazioaren gorputza honako aldibereko asignazioa da:

     ```
     (x,y,v) := (x+y,x,v-1);
     ```

     Aldibereko asignazio hau inplementatzeko, aldagai laguntzaile bat erabili behar dugu:

     ```
     lag := x;
     x := x+y;
     y := lag;
     v := v-1;
     ```

Horrenbestez, urrats guztiak bildurik, hona hemen lortu dugun funtzio iteratiboa:

```
function fib_it(n :  Integer) return Integer is
   x,y,v,lag :  Integer;
begin      { n ≥ 0 }
   x := 0;
   y := 1;
   v := n;
   while not v = 0 loop        INB ≡ { fib(n) = x × fib(v + 1) + y × fib(v) }
                               E ≡ v
      lag := x;
      x := x+y;
      y := lag;
      v := v-1;
   end loop;
   return x;
end fib_it;
{ fib_it(n) = fib(n) }
```