

5. Gaia

Datu-Mota Abstraktuen ekuazio bidezko espezifikazioa

5.1. Sarrera.....	3
5.1.1. Helburua	3
5.1.2. Haskell.....	3
5.2. Haskell-en aurredefinituta dauden oinarritzko motak	4
5.2.1. Bool	4
5.2.2. Int.....	4
5.2.3. Integer.....	4
5.2.4. Float, Double	5
5.2.5. Char	5
5.2.6. Oinarritzko motentzako erlazio-eragileak	5
5.2.7. Oinarritzko motentzat eragiketa berrien definizioa	5
5.3. Zerrendak.....	7
5.3.1. Eragiketa eraikitzaileak zerrendentzat.....	7
5.3.2. Eragiketa ez-eraikitzaile batzuk zerrendentzat	8
5.3.4. Zerrendentzako eragile erlazio-eragileak	14
5.3.5. Propietateen frogapen formula: Indukzioa	15
5.3.6. Espresioen sinplifikazioa.....	22
5.4. Pilak.....	23
5.4.1. Eragiketa eraikitzaileak pilentzat.....	23
5.4.2. Eragiketa ez-eraikitzaile batzuk pilentzat.....	25
5.5. Ilarak.....	27
5.5.1. Eragiketa eraikitzaileak ilara datu-motarentzat	27
5.5.2. Eragiketa ez-eraikitzaile batzuk ilarentzat.....	29
5.6. Zuhaitz bitarrak.....	31
5.6.1. Eragiketa eraikitzaileak zuhaitz bitarrentzat.....	34
5.6.2. Eragiketa ez-eraikitzaile batzuk zuhaitz bitarrentzat.....	37
5.7. Datu-mota abstraktuak nahasian dituzten adibideak	43
5.8. Modularitatea Haskell-en	45
5.8.1. Bi moduluren definizioa aurreko ataletan emandako eragiketak erabiliz	45
5.9. Oharrak.....	47
5.9.1. Maiuskula eta minuskulen erabilera izenetan.....	47
5.9.2. Azalpenak ipintzeko bi era	47

5.1. Sarrera

5.1.1. Helburua

Ekuazio bidezko espezifikazioaren (edo aljebra bidezko espezifikazioaren) teknika erabiliz Datu-Mota Abstraktu berriak (DMA berriak) nola definitzen diren aztertzea da gai honetako helburua.

Datu-mota berri bat definitzeko jarraian aipatzen direnak zehaztu behar dira

- Mota horretako balioak zein diren.
- Mota horretako balioekin buru daitezkeen eragiketak.
- Mota horrentzat definitutako eragiketek betetzen dituzten propietateak.

Adibidez, zenbaki osoei dagokien datu-motaren kasuan, balioak zenbaki osoak beraiek dira (... , -2, -1, 0, 1, 2, 3, ...), eragiketak batuketa, kenketa eta abar dira eta eragiketa horien propietateak batuketa trukakorra eta elkarkorra dela, kenketa ez dela trukakorra, biderketa banakorra dela batuketarekiko eta abar esaten digute.

Datu-mota bateko balioak zein diren zehazteko eta balio horien gaineko eragiketak definitzeko gehienetan errekurtsibitatea erabiltzen da eta propietateak frogatzerakoan aldiz tresna garrantzitsuena indukzioa da.

5.1.2. Haskell

Haskell programazio-lengoaia funtzioetan oinarritzen da. Beraz, *funtzio bidezko programazioa* bezala ezagutzen da Haskell (eta bera bezalako beste lengoaia batzuk) erabiliz burutzen den programazioa. Haskell lengoaiaren funtzio guztiak "garbiak" dira, matematikako zentzuan: funtzioek ez dute albo-ondoriorik eragiten (ezta Sarrera/Irteeraz arduratzen diren funtzioek ere). Hori dela eta, Haskell lengoaia erabiliz eraikitzen diren espresio edo adierazpen denak "gardenak" dira ("aipamen-gardentasuna" bezala ezagutzen den propietatea dute).

Haskell lengoaiaren programa bat funtzio multzo bat izan ohi da. Funtzio bakoitza era independentean exekuta daiteke eta baita beste funtzio batetik deituz ere (kasu bietan parametro formalen ordeztu argumentu errealak ipiniz).

Haskell-en web orri ofiziala <http://haskell.org> da eta bertatik sistema eragile desberdinetarako Haskell plataforma instalatu daiteke: <https://www.haskell.org/platform/>. Bestalde <https://www.haskell.org/downloads> orrian beste aukera batzuk ere badira. Gainera GHC konpilatzailea ere eskura daiteke <http://www.haskell.org/ghc/> helbidean.

Haskell lengoaiaren programak idazteko, notepad++ erabil daiteke editore bezala. Editore hau <http://notepad-plus-plus.org> helbidean dago eskuragarri.

5.2. Haskell-en aurredefinituta dauden oinarrizko motak

5.2.1. Bool

- Balioak: {True, False}
- Eragiketak:
 - ✓ && (and)
 - ✓ || (or)
 - ✓ not (not)
- Propietateak (ϕ , ψ eta γ espresio boolearrak izanda):
 - ✓ $\phi \ \&\& \ \psi \equiv \psi \ \&\& \ \phi$ (&& trukakorra da)
 - ✓ $\phi \ || \ \psi \equiv \psi \ || \ \phi$ (|| trukakorra da)
 - ✓ $\text{not}(\phi \ \&\& \ \psi) \equiv (\text{not} \ \phi) \ || \ (\text{not} \ \psi)$ (De Morgan-en legea)
 - ✓ $\text{not}(\phi \ || \ \psi) \equiv (\text{not} \ \phi) \ \&\& \ (\text{not} \ \psi)$ (De Morgan-en legea)
 - ✓ $\phi \ \&\& \ (\psi \ || \ \gamma) \equiv (\phi \ \&\& \ \psi) \ || \ (\phi \ \&\& \ \gamma)$ (Banakortasuna)
 - ✓ $\phi \ || \ (\psi \ \&\& \ \gamma) \equiv (\phi \ || \ \psi) \ \&\& \ (\phi \ || \ \gamma)$ (Banakortasuna)
 - ✓ $\text{False} \ || \ \phi \equiv \phi$ (False || eragilearentzat elementu neutroa da)
 - ✓ $\text{False} \ \&\& \ \phi \equiv \text{False}$ (False && eragilearentzat elementu nulua da)
 - ✓ $\text{True} \ || \ \phi \equiv \text{True}$ (True || eragilearentzat elementu nulua da)
 - ✓ $\text{True} \ \&\& \ \phi \equiv \phi$ (True && eragilearentzat elementu neutroa da)
 - ✓ ...

5.2.2. Int

- Balioak: tarte mugatu bateko zenbaki osoak (minBound, maxBound)
- Eragile aritmetikoak:
 - ✓ - (zeinu negatiboa)
 - ✓ + (batuketa)
 - ✓ - (kenketa)
 - ✓ * (biderketa)
 - ✓ ^ (berredura, $2^3 = 8$)
 - ✓ div (zatiketa osoa; idazteko bi era: $16 \div 3 = 5$ edo $16 \div 3 = 5$, eta `karakterea P letraren ondoan dagoen azentu-marka da)
 - ✓ mod (zatiketa osoaren hondarra; idazteko bi era: $16 \bmod 3 = 1$ edo $16 \bmod 3 = 1$, eta `karakterea P letraren ondoan dagoen azentu-marka da)
- Propietateak:

✓ $x + y = y + x$	Trukakortasuna
✓ $x + (y + z) = (x + y) + z$	Elkarkortasuna
✓ $x * (y + z) = (x * y) + (x * z)$	Banakortasuna
✓ $0 + x = x$	0 neutroa da batuketarentzat
✓ $1 * x = x$	1 neutroa da biderketarentzat
✓ $0 * x = 0$	0 nulua da biderketarentzat
✓ ...	

5.2.3. Integer

- Balioak: zenbaki osoak inolako mugarik gabe
- Eragile aritmetikoak: Int kasukoak

- Propietateak: Int kasukoak

5.2.4. Float, Double

- Balioak: zenbaki errealak
- Eragile aritmetikoak: Ez ditugu erabiliko
- Propietateak: Int kasukoan antzekoak

5.2.5. Char

- Balioak: karaktereak ('a', ..., 'z', 'A', ..., 'Z', ...) Oren teklaren ondoan dagoen teklako apostrofoa erabiliz
- Eragileak: Ez ditugu erabiliko
- Propietateak: Ez ditugu aztertuko

5.2.6. Oinarrizko motentzako erlazio-eragileak

Jarraian erakusten diren erlazio-eragileak aipatu diren oinarrizko mota denentzat balio dute:

- ✓ == (berdin)
- ✓ /= (ezberdin)
- ✓ > (handiagoa)
- ✓ >= (handiagoa edo berdina)
- ✓ < (txikiagoa)
- ✓ <= (txikiagoa edo berdina)

5.2.7. Oinarrizko motentzat eragiketa berrien definizioa

Lehenago azaldu diren eragiketak erabiltzeaz gain eragiketa berriak ere defini eta erabil daitezke. Eragiketak definitzerakoan ekuazio bakoitzari zenbaki bat egokituko diogu geroago ekuazio hori aipatu ahal izateko. Hiru adibide emango dira jarraian:

- bikoitia: zenbaki oso bat emanda, True itzuliko du bikoitia baldin bada eta False bakoitia baldin bada.

Eragiketaren **mota**:

bikoitia:: (Int) -> Bool

Eragiketa definitzen duen **ekuazioa**:

$$\text{bikoitia}(x) = (x \text{ `mod` } 2) == 0 \quad (1)$$

bikoitia funtzioak x balioarentzat $x \text{ `mod` } 2$ eta 0 alderatzearen balioa itzuliko duela adierazten da ekuazio horren bidez (konparazioaren emaitza True edo False izango da).

Eragiketa bera definitzeko beste aukera bat honako hau izango litzateke:

$$\begin{aligned} \text{bikoitia}(x) \\ &| (x \text{ `mod` } 2) == 0 &= \text{True} & (1) \\ &| \text{otherwise} &= \text{False} & (2) \end{aligned}$$

Definizio hori honela ulertu beharko genuke:

$x \bmod 2$ eta 0 berdina bada True itzuli eta bestela False itzuli.

- **bakoitia**: zenbaki oso bat emanda, True itzuliko du zenbaki hori bakoitia bada eta False itzuliko du zenbaki hori bikoitia bada.

Eragiketaren **mota**:

bakoitia:: (Int) -> Bool

Eragiketa definitzen duen **ekuazioa**:

bakoitia(x) = not(bikoitia(x)) (1)

Kasu honetan, x zenbaki oso bat emanda, *bakoitia* funtzioak x zenbaki horrentzat itzuliko duena *bikoitia* funtzioak x zenbaki horrentzat itzuliko lukeenaren aurkakoa dela adierazten da. Beraz, hasteko *bikoitia* funtzioak x balioarentzat itzultzen duena kalkulatu da eta gero balio horren aurkakoa hartuko da *not* eragilearen bidez.

Adibide honetan ikusten den bezala, aurretik definituta dauden funtzioak ere erabil daitezke funtzio berri bat definitzerakoan.

- **hand3**: hiru zenbaki oso emanda handiena itzuliko du

Eragiketaren **mota**:

hand3:: (Int, Int, Int) -> Int

Eragiketa definitzen duten **ekuazioak**:

hand3(x, y, z)

$x \geq y \ \&\& \ x \geq z$	= x	(1)
$y > x \ \&\& \ y \geq z$	= y	(2)
otherwise	= z	(3)

Definizio hori honela ulertu beharko genuke:

$x \geq y \ \&\& \ x \geq z$ betetzen bada, orduan x balioa itzuli, bestela $y > x \ \&\& \ y \geq z$ betetzen bada, orduan y itzuli eta bestela z itzuli.

Adibide honetan bezala baldintza-zerrenda bat agertzen denean, baldintzak goitik behera aztertuko dira Haskell-en eta betetzen den lehenengo baldintzaren kasuari dagokion emaitza itzuliko da.

5.3. Zerrendak

Zerrendak honako era honetan adierazten dira: [2, 7, 35, -8, 0]

Zerrenda hori zenbaki osoz osatuta dago eta bere mota [Int] da. Zerrenda hutsa honako era honetan adierazten da: []

Zerrenden mota parametrodun mota da, [t], eta t parametroaren arabera balio boolearrez osatutako zerrendak, zenbaki osoz osatutako zerrendak eta abar eduki ditzakegu. Beraz aurretik definitutako edozein mota erabiliz mota horretako elementuz osatutako zerrendak defini daitezke baina zerrenda bateko elementu denak mota berekoak izan behar dute.

Adibideak:

[True, True, True, False, True] Bere mota [Bool] da (boolearrez osatutako zerrenda)
[[0, 5], [], [77, -50, 3]] Bere mota [[Int]] da (zenbaki osoez osatutako zerrendez eratutako zerrenda)

5.3.1. Eragiketa eraikitzaileak zerrendentzat

Datu-mota bati dagozkion eragiketa eraikitzaileak mota horretako balioak zein diren adierazteko dira.

Edozein zerrenda honako bi eragile hauek erabiliz eraiki daitezke:

- [] **Zerrenda hutsa sortu**
- : **Ezkerretik elementu bat erantsi**

Lehenengo eragiketak zerrenda hutsa sortzeko balio du eta bigarrenak ezkerretik elementuak banan-banan erantsiz edo sartuz joateko balio du.

Esate baterako [6, 10] zerrenda honako era honetan eraikiko litzateke: 6:10:[]

Hasteko zerrenda hutsa sortuko litzateke, gero 10 zenbakia gehitu edo erantsiko litzaioke zerrenda hutsari eta bukatzeko 6 balioa gehituko litzaioke dagoeneko 10 zenbakia duen zerrendari.

[] → [10] → [6, 10]

[2, 7, 35, -8, 0] zerrenda honako era honetan eraikiko litzateke: 2:7:35:-8:0:[]

Haskell-entzat [6,10] eta 6:10:[] baliokideak dira, biak onartzen ditu, baina ekuazio bidezko espezifikazioa erabiliz eragiketa berriak definitzeko eta eragiketa berri horien propietateak frogatzeko bigarren aukera egokiagoa da.

Edozein zerrenda jarraian aipatzen diren bi egitura hauetakoren batera egokitzen da:

- ✓ [] (zerrenda hutsa da)
- ✓ x:s (ez da zerrenda hutsa eta bere lehenengo elementua x da eta beste elementu denek s azpi-zerrenda osatzen dute) (x eta s-ren ordezkariak beste izen batzuk ere erabil daitezke).

[2, 7, 35, -8, 0] zerrenda 2:7:35:-8:0:[] eran idatz daiteke

x: s

s azpizerrenda 7:35:-8:0:[] zerrenda da

Zerrenda batek bi elementu edo gehiago ditunean

$x:z:s$

erakoa dela ere esan daiteke, komeni bada. Lehenengo elementua x izango da, bigarrena z eta gainontzeko elementuek s azpizerrenda osatuko dute.

[2, 7, 35, -8, 0] es $2:7:35:-8:0:[]$
 $x:z:s$

Eragiketa eraikitzaileen motak honako hauek dira:

$[] :: [t]$
 $: :: (t, [t]) \rightarrow [t]$

Lehenengo eragiketaren motak, hau da, $[]$ eragiketaren motak, eragiketa horrek geuk nahi dugun motako (t motako) zerrenda hutsa sortuko duela adierazten du. $[]$ eragileak sarrerako argumenturik ez duenez jasotzen, \rightarrow ez da ipini behar mota zehazterakoan.

Bigarren eragiketaren kasuan, $:$ eragileari t motako elementu bat eta t motako zerrenda bat emanda, t motako zerrenda bat itzuliko duela adierazten da. Eragiketen motak eta elementuen motak ematerakoan $::$ notazioa erabiltzen da. Funtzioetan sarrerako datuak parentesi artean joango dira komaz bereizita eta emaitza \rightarrow sinboloen ondoren joango da. Sarrerako daturik ez badago, emaitzaren mota bakarrik emango da, $[]$ eragilearen kasuan bezala. Funtzioen kasuan gerta daiteke sarrerako daturik ez egotea, datu bakarra egotea edo datu bat baino gehiago egotea, baina beti emaitza bat egongo da.

5.3.2. Eragiketa ez-eraikitzaile batzuk zerrendentzat

Jarraian zerrendekin kalkuluak burutzeko balio duten eragiketa batzuk definituko dira. Funtzio bat edo eragiketa bat definitzeko, funtzioaren mota eta funtzioak zer egiten duen zehazten duten ekuazioak eman behar dira.

- **leh:** Zerrenda bat emanda, zerrendako lehenengo elementua itzuliko du. Zerrenda hutsa baldin bada, errorea sortuko da.

Adibideak:

$\text{leh}([4, 7, 8]) = 4$
 $\text{leh}([False, True]) = False$
 $\text{leh}([0, 5], [], [77, -50, 3]) = [0, 5]$

Eragiketaren **mota:**

$\text{leh} :: ([t]) \rightarrow t$

Eragiketa definitzen duten **ekuazioak:**

$\text{leh}([]) = \text{error "Zerrenda hutsa. Ez dago lehenengo elementurik."}$ (1)
 $\text{leh}(x:s) = x$ (2)

Errore-mezu bat aurkezteko Haskell-eko *error* funtzioa erabiltzen da eta zein mezu aurkeztu nahi den zehaztu beharko da komatxo bikoitzen artean.

- **hond**: Zerrenda bat emanda, zerrendako lehenengo elementua kenduz lortzen den zerrenda itzuliko du. Zerrenda hutsa baldin bada, errorea sortuko da.

Adibideak:

```
hond([4, 7, 8]) = [7, 8]
hond([False, True]) = [True]
hond([[0, 5], [], [77, -50, 3]]) = [[], [77, -50, 3]]
```

Eragiketaren **mota**:

```
hond:: ([t]) -> [t]
```

Eragiketa definitzen duten **ekuazioak**:

```
hond([]) = error "Zerrenda hutsa. Ez dago hondarrik." (1)
```

```
hond(x:s) = s (2)
```

Zerrenda hutsari dagokion errore-mezua aurkezteko Haskell-eko *error* funtzioa erabili da eta aurkeztu nahi den mezua zein den zehaztu da komatxo bikoitzen artean.

- **hutsa da**: Zerrenda bat emanda, True itzuliko du zerrenda hutsa baldin bada eta False itzuliko du zerrenda hutsa ez bada.

Adibideak:

```
hutsa_da([]) = True
hutsa_da([4, 7, 8]) = False
hutsa_da([False, True]) = False
```

Eragiketaren **mota**:

```
hutsa_da:: ([t]) -> Bool
```

Eragiketa definitzen duten **ekuazioak**:

```
hutsa_da([]) = True (1)
```

```
hutsa_da(x:s) = False (2)
```

Emandako zerrenda hutsa baldin bada (hau da, [] erakoa baldin bada), orduan True itzuliko da. Emandako zerrenda hutsa ez bada (hau da, x:s erakoa baldin bada, x zerrendako lehenengo elementua izanda eta s zerrendako gainontzeko elementuez osatutako azpizerrenda izanda), orduan False itzuliko da.

- **badago:** t motako elementu bat eta t motako zerrenda bat emanda, True balioa itzuliko du elementua zerrendan baldin badago eta False itzuliko du ez badago.

Adibideak:

```
badago(8, []) = False
badago(8, [4, 8, 7, 8]) = True
badago(False, [False, True, True]) = True
badago([10, 9], [[0, 5], [], [77, -50, 3]]) = False
```

Eragiketaren **mota:**

badago:: (t, [t]) -> Bool

Eragiketa definitzen duten **ekuazioak:**

badago(x, []) = False (1)

```
badago(x, y:s)
| x == y      = True      (2)
```

```
| x /= y      = badago(x, s) (3)
```

Ekuazio horien bitartez x elementua zerrenda batean agertzen al den ala ez erabakitzen da. Zerrenda hutsa baldin bada (hau da, [] erakoa baldin bada), x ez da hor egongo eta False itzuliko da zuzenean. Zerrenda hutsa ez bada, (hau da, y:s erakoa baldin bada, zerrendako lehenengo elementua y dela eta zerrendako beste elementu denez osatutako azpizerrenda s dela kontsideratuz) x balioa y:s zerrendako lehenengo elementuaren berdina baldin bada (hau da, x balioa y-ren berdina baldin bada) erantzuna True izango da zuzenean baina x balioa y:s zerrendako lehenengo elementuaren berdina ez bada (hau da x eta y berdinak ez badira), orduan x elementua s azpizerrendan ba al dagoen begiratu beharko da.

- **luzera:** t motako zerrenda bat emanda, elementu-kopurua (zerrenda horretan zenbat elementu dauden) itzuliko du.

Adibideak:

```
luzera([]) = 0
luzera([5, 8, 7, 8]) = 4
luzera([False, True, True]) = 3
luzera([[0, 5], [], [77, -50, 3]]) = 3
```

Eragiketaren **mota:**

luzera:: ([t]) -> Int

Eragiketa definitzen duten **ekuazioak:**

luzera([]) = 0 (1)

luzera(x:r) = 1 + luzera(r) (2)

Emandako zerrenda hutsa baldin bada (hau da, [] erakoa baldin bada), orduan luzera edo elementu-kopurua 0 izango da. Emandako zerrenda hutsa ez bada (hau da, x:r erakoa baldin bada), zerrendak gutxienez x elementua du eta gero r azpizerrendako elementu-kopurua zenbatu beharko da.

- **bikop**: Int motako zerrenda bat emanda, zerrendan zenbat elementu bikoiti dauden itzuliko du.

Adibideak:

$\text{bikop}([]) = 0$
 $\text{bikop}([5, 8, 7, 8]) = 2$
 $\text{bikop}([7, 11, 9]) = 0$
 $\text{bikop}([0, 3, 3, 5]) = 1$

Eragiketaren **mota**:

$\text{bikop} :: ([\text{Int}]) \rightarrow \text{Int}$

Eragiketa definitzen duten **ekuazioak**:

$\text{bikop}([]) = 0$ (1)

$\text{bikop}(x:r)$

| $\text{bikoitia}(x) = 1 + \text{bikop}(r)$ (2)

| $\text{bakoitia}(x) = \text{bikop}(r)$ (3)

Emandako zerrenda hutsa baldin bada (hau da, $[]$ erakoa baldin bada), 0 elementu bikoiti daude zerrenda horretan. Emandako zerrenda hutsa ez bada, hau da, adibidez $x:r$ erakoa baldin bada (zerrendako horretako lehenengo elementua x dela eta zerrendako beste elementu denez osatutako azpizerrenda r dela kontsideratuz), zerrendako lehenengo elementua bikoitia baldin bada (hau da, x bikoitia baldin bada) bikoiti-kopurua 1 gehi " r azpizerrendako bikoiti-kopurua" izango da. Baina zerrendako lehenengo elementua bakoitia baldin bada (hau da, x bakoitia baldin bada) bikoiti-kopurua " r azpizerrendako bikoiti-kopurua" izango da, x ez da zenbatu behar eta, ez baita bikoitia.

bikop funtzioaren definizioan lehendik definituta ditugun *bikoitia* eta *bakoitia* funtzioak erabili dira. Funtzio berriak definitzerakoan lehendik definituta dauden funtzioak erabil daitezke laguntzaile bezala.

- **++**: t motako bi zerrenda emanda, bata besteari erantsiz edo biak elkartuz lortzen den zerrenda itzuliko du.

Adibideak:

$$[] ++ [] = []$$

$$[4, 8, 7, 8] ++ [] = [4, 8, 7, 8]$$

$$[1, 8] ++ [4, 8, 7] = [1, 8, 4, 8, 7]$$

$$[\text{False}, \text{True}, \text{True}] ++ [\text{False}, \text{False}] = [\text{False}, \text{True}, \text{True}, \text{False}, \text{False}]$$
Eragiketaren mota:

$$++ :: ([t], [t]) \rightarrow [t]$$
Eragiketa definitzen duten ekuazioak:

$$[] ++ s = s \quad (1)$$

$$(x:r) ++ s = x:(r ++ s) \quad (2)$$

Bigarren ekuazioak x:r eta s zerrendak elkartzeko lehenengo r eta s zerrendak elkartu eta gero r ++ s zerrenda berriari x elementua ezkerretik gehitu behar zaiola dio.

- **tartekatu**: t motako bi zerrenda emanda, bi zerrenda horietako elementuak tartekatuz lortzen den zerrenda itzuliko du funtzio honek. Lehenengo zerrendako lehenengo elementua zerrenda berriko lehenengo elementua izango da, bigarren zerrendako lehenengo elementua zerrenda berriko bigarren elementua izango da eta abar. Hasierako zerrenda biak luzera berekoak ez badira, errorea gertatuko da (errore-mezua aurkeztuko da).

Adibidea:

tartekatu([7, 5, 4], [8, 2, 0]) = [7, 8, 5, 2, 4, 0]

Eragiketaren **mota**:

tartekatu:: ([t], [t]) -> [t]

Eragiketa definitzen duten **ekuazioak**:

tartekatu([], s)

| not hutsa_da(s) = error "Luzera desberdineko zerrendak." (1)

| otherwise = [] (2)

tartekatu(x:r, s)

| luzera(x:r) /= luzera(s) = error "Luzera desberdineko zerrendak." (3)

| otherwise = x:(leh(s): tartekatu(r, hond(s))) (4)

Kasu honetan 4 ekuazio eman dira *tartekatu* funtzioa definitzeko.

Lehenengo bi ekuazioetan, lehenengo zerrenda hutsa denean ([]) erakoa denean zer egin behar den zehazten da. Bigarren zerrenda (s zerrenda) hutsa ez bada, zerrenda bien luzera desberdina izango denez, errore-mezua aurkeztuko da eta bestela, hau da, bigarren zerrendaren luzera [] zerrendaren luzeraren berdina denean, zerrenda biak hutsak direnez emaitza bezala ere zerrenda hutsa itzuliko da.

Beste bi ekuazioetan lehenengo zerrenda hutsa ez denean (adibidez x:r erakoa denean, zerrendako lehenengo elementua x dela eta zerrendako gainontzeko elementuez osatutako azpizerrenda r dela kontsideratuz), zer egin behar den zehazten da. Hor (3) ekuazioan x:r zerrendaren luzera eta s zerrendaren luzera desberdinak baldin badira errore-mezua aurkeztuko dela adierazten da. Laugarren ekuazioan x:r eta s zerrendek luzera bera dutenean bi zerrenda horietako elementuak tartekatuz osatutako zerrenda nola eraikitzen den zehazten da. Zerrenda berriko lehenengo elementua x:r zerrendako lehenengo elementua izango da (x elementua). Zerrenda berriko bigarren elementua s zerrendako lehenengo elementua izango da, baina elementu horrek izen berezirik ez duenez, *leh(s)* bezala adierazi behar da. Zerrenda berriaren gainontzeko zatia kalkulatzeko, lehenengo zerrendako (x:r zerrendako) gainontzeko elementuak, hau da, r azpizerrendako elementuak eta bigarren zerrendako (s zerrendako) gainontzeko elementuak tartekatu beharko dira. Baina s zerrendako gainontzeko elementuez osatutako azpizerrendari izenik ez diogunez eman, *hond(s)* bezala adierazi beharko da azpizerrenda hori.

- **tartekatu2**: t motako bi zerrenda emanda, bi zerrenda horietako elementuak tartekatuz lortzen den zerrenda itzuliko du baina kasu honetan lehenengo zerrendako lehenengo elementua zerrenda berriko bigarren elementua izango da, bigarren zerrendako lehenengo elementua zerrenda berriko lehenengo elementua izango da eta abar. Hasierako zerrenda biak luzera berekoak ez badira, errorea gertatuko da (errore-mezua aurkeztuko da).

Adibidea:

$\text{tartekatu2}([7, 5, 4], [8, 2, 0]) = [8, 7, 2, 5, 0, 4]$

Eragiketaren **mota**:

$\text{tartekatu2}:: ([t], [t]) \rightarrow [t]$

Eragiketa definitzen duten **ekuazioak**:

$\text{tartekatu2}([], s)$

| not hutsa_da(s) = error "Luzera desberdineko zerrendak." (1)

| otherwise = [] (2)

$\text{tartekatu2}(x:r, s)$

| luzera(x:r) /= luzera(s) = error "Luzera desberdineko zerrendak." (3)

| otherwise = leh(s):(x: tartekatu2(r, hond(s))) (4)

Ekuazio hauek ematerakoan aurreko adibideko, hau da, *tartekatu* izeneko funtzioaren kasuko ideia bera jarraitu da, baina (4) ekuazioan zerrenda berria eraikitzerakoan elementuak aurkako ordenan ipini dira lortu nahi den zerrenda eraiki ahal izateko. Hori da hain zuzen ere *tartekatu* eta *tartekatu2* izeneko funtzioen arteko desberdintasuna.

5.3.4. Zerrendentzako eragile erlazio-eragileak

Oinarrizko datu-moten atalean aipatutako erlazio-eragile zerrendekin ere erabil daitezke.

Zerrenda bat beste bat baino txikiagoa edo handiagoa al den erabakitzeke hitzak alfabetikoki ordenatzeko erabiltzen den teknika bera jarraitzen da.

Esate baterako [3, 3, 4] zerrenda [5, 1] zerrenda baino txikiagoa da 3 zenbakia 5 baino txikiagoa delako.

5.3.5. Propietateen frogapen formal: Indukzioa

Zerrendentzat definitutako eragiketek betetzen dituzten propietateak indukzioa (zerrenden egituraren oinarritutako indukzioa) erabiliz frogatu ohi dira gehienetan. Eragiketek propietate batzuk betetzen dituztela nola frogatzen den ikusteko, adibide batzuk azalduko ditugu atal honetan.

1. adibidea

Edozein s zerrendak honako propietate hau beteko du:

$$\text{luzera}(s) \geq 0$$

Indukzioa s zerrendarekiko burutuko da.

- Oinarritzko kasua: s zerrenda hutsa da, hau da, $s = []$

$$\text{luzera}([]) \geq 0?$$

$\text{luzera}([]) = 0$ (luzera eragiketaren lehenengo ekuazioagatik)
Beraz $\text{luzera}(s) \geq 0$ bete egiten da s zerrenda hutsa denean.

- Kasu induktiboa edo orokorra: s ez da zerrenda hutsa, hau da, $s = x:r$

$$\text{luzera}(x:r) \geq 0?$$

Kasu honetan r zerrendaren luzera 0ren berdina edo handiagoa dela suposatuz $x:r$ zerrendaren luzera 0ren berdina edo handiagoa dela frogatu behar dugu.

- **Indukzio-hipotesia** (ih): r zerrendak propietatea bete egiten duela suposatuko dugu, hau da, $\text{luzera}(r) \geq 0$ dela.
- **Indukzio-urratsa**: r zerrendak propietatea betetzen duela suposatuz $x:r$ zerrendak ere propietatea bete egiten duela frogatu behar da.

$$\begin{aligned} \text{luzera}(x:r) &= (\text{luzera-ren bigarren ekuazioagatik}) \\ &= (1 + \text{luzera}(r)) \end{aligned}$$

$(1 + \text{luzera}(r)) \geq 0$ al da? Bai. Indukzio hipotesiagatik badakigu $\text{luzera}(r)$ 0 edo handiagoa dela eta ondorioz $1 + \text{luzera}(r)$ balioa 1 edo handiagoa izango da eta beraz ≥ 0 ere izango da.

Hasierako galdera eraldatzean oinarritzen den beste era honetara ere egin daiteke frogapena:

$\text{luzera}(x:r) \geq 0?$ (luzera-ren bigarren ekuazioa aplikatuko da orain)
 $1 + \text{luzera}(r) \geq 0?$ (inekuazioaren alde bakoitzari 1 kenduko zaio orain)
 $1 - 1 + \text{luzera}(r) \geq 0 - 1?$ (orain eragiketak burutuko dira)
 $\text{luzera}(r) \geq -1?$ Erantzuna "Bai" da, izan ere indukzio-hipotesiagatik badakigu $\text{luzera}(r) \geq 0$ betetzen dela eta, ondorioz, $\text{luzera}(r) \geq -1$ ere beteko da.

2. adibidea

Har ditzagun s eta r edozein bi zerrenda, honako propietate hau betetzen dutela frogatuko dugu indukzioaren bidez:

$$\text{luzera}(s ++ r) = \text{luzera}(s) + \text{luzera}(r)$$

Indukzioa s zerrendari aplikatuko zaio.

- Oinarrizko kasua: s zerrenda hutsa da, hau da, $s = []$

$$\text{luzera}([] ++ r) = \text{luzera}([]) + \text{luzera}(r)?$$

Ekuazioaren alde biek balio bera dutela frogatu behar da.

$$\begin{aligned} \text{luzera}([] ++ r) &= & (\text{++ eragilearen 1. ekuazioagatik}) \\ &= \text{luzera}(r) \end{aligned}$$

$$\begin{aligned} \text{luzera}([]) + \text{luzera}(r) &= & (\text{luzera eragiketaren 1. ekuazioagatik}) \\ &= 0 + \text{luzera}(r) = & (\text{batuketarentzat 0 neutroa delako}) \\ &= \text{luzera}(r) \end{aligned}$$

Ekuazioaren alde bietan emaitza bera lortu da eta, ondorioz, s zerrenda hutsa denean propietatea bete egiten dela frogatu dugu.

- Kasu induktiboa: s ez da zerrenda hutsa, hau da, $s = x:w$

Honako hau frogatu behar dugu

$$\text{luzera}((x:w) ++ r) = \text{luzera}(x:w) + \text{luzera}(r).$$

- **Indukzio-hipotesia** (ih): w eta r -rentzat propietatea bete egiten dela suposatuko dugu

$$\text{luzera}(w ++ r) = \text{luzera}(w) + \text{luzera}(r)$$

- **Indukzio-urratsa**: w eta r zerrendentzat propietatea bete egiten dela suposatuz, $x:w$ eta r zerrendentzat ere bete egiten dela frogatuko dugu. Horretarako ekuazioaren alde biek balio bera dutela egiaztatu beharko da:

$$\begin{aligned} \text{luzera}((x:w) ++ r) &= & (\text{++ eragilearen 2. ekuazioa erabiliz}) \\ &= \text{luzera}(x:(w ++ r)) = & (\text{luzera eragiketaren 2. ekuazioa erabiliz}) \\ &= 1 + \text{luzera}(w ++ r) \end{aligned}$$

$$\begin{aligned} \text{luzera}(x:w) + \text{luzera}(r) &= & (\text{luzera eragiketaren 2. ekuazioa erabiliz}) \\ &= (1 + \text{luzera}(w)) + \text{luzera}(r) = & (\text{batuketa elkarkorra delako}) \\ &= 1 + (\text{luzera}(w) + \text{luzera}(r)) = & (\text{indukzio-hipotesiagatik}) \\ &= 1 + \text{luzera}(w ++ r) \end{aligned}$$

Ekuazioaren alde bietan gauza bera lortu da eta ondorioz s zerrenda $x:w$ erakoa denean ere propietatea bete egiten dela frogatu dugu.

Oinarrizko kasuan s zerrenda hutsa denean ($[]$ erakoa denean) propietatea bete egiten dela frogatu da eta kasu induktiboan s hutsa ez denean ere (hau da, $x:w$ erakoa denean ere) propietatea bete egiten dela frogatu da. Beraz propietatea beti betetzen dela frogatu dugu, s eta r edozein bi zerrenda izanda.

3. adibidea

Edozein balio (x) eta edozein bi zerrenda (s eta r) hartuta, jarraian adierazten den propietatea betetzen dela frogatuko dugu adibide honetan:

$$\text{badago}(x, s ++ r) = \text{badago}(x, s) \parallel \text{badago}(x, r)$$

Indukzioa s zerrendarekiko egingo da.

- Oinarrizko kasua: s zerrenda hutsa da, hau da, $s = []$

$$\text{badago}(x, [] ++ r) = \text{badago}(x, []) \parallel \text{badago}(x, r)?$$

Ekuazioaren alde biek balio bera dutela egiaztatu behar da.

$$\begin{aligned} \text{badago}(x, [] ++ r) &= \quad (\text{++ eragilearen 1. ekuazioagatik}) \\ &= \text{badago}(x, r) \end{aligned}$$

$$\begin{aligned} \text{badago}(x, []) \parallel \text{badago}(x, r) &= \quad (\text{badago eragiketaren 1. ekuazioagatik}) \\ &= \text{False} \parallel \text{badago}(x, r) = \quad (\parallel \text{eragilearentzat False neutroa delako}) \\ &= \text{badago}(x, r) \end{aligned}$$

Ekuazioaren alde bietan emaitza bera lortu da eta ondorioz s zerrenda hutsa denean propietatea bete egiten dela frogatu dugu.

- Kasu induktiboa: s ez da zerrenda hutsa, hau da, $s = z:w$

Honako hau frogatu behar da

$$\text{badago}(x, (z:w) ++ r) = \text{badago}(x, z:w) \parallel \text{badago}(x, r)$$

- **Indukzio-hipotesia** (ih): w eta r zerrendentzat propietatea bete egiten dela suposatuko dugu

$$\text{badago}(x, w ++ r) = \text{badago}(x, w) \parallel \text{badago}(x, r)$$

- **Indukzio-urratsa**: w eta r zerrendek propietatea bete egiten dutela suposatuz, $z:w$ eta r zerrendek ere bete egiten dutela frogatu behar da. Horretarako ekuazioaren alde biek balio bera dutela egiaztatu behar da. Kasu honetan, *badago* eragiketa definitzerakoan bigarren parametro bezala hutsa ez den zerrenda agertzen denean bi azpikasu bereizi dira eta, ondorioz, bi ekuazio eman behar izan dira. Azpikasu bakoitzeko froga aparte egin beharko da.

(2) $x = z$

$$\begin{aligned}
 \text{badago}(x, (z:w) \text{ ++ } r) &= && (\text{++ eragilearen 2. ekuazioa erabiliz}) \\
 = \underline{\text{badago}}(x, z:(w \text{ ++ } r)) &= && (\underline{\text{badago}} \text{ eragilearen 2. ekuazioagatik}) \\
 = \mathbf{True}
 \end{aligned}$$

$$\begin{aligned}
 \text{badago}(x, z:w) \parallel \text{badago}(x, r) &= && (\underline{\text{badago}} \text{ eragilearen 2. ekuazioagatik}) \\
 = \text{True} \parallel \text{badago}(x, r) &= && (\text{True} \parallel \varphi \equiv \text{True delako}) \\
 = \mathbf{True}
 \end{aligned}$$

Ekuazioaren alde bietan emaitza bera lortu da eta ondorioz s zerrenda z:w erakoa denean eta $x = z$ denean propietatea bete egiten dela frogatu dugu.

(3) $x \neq z$

$$\begin{aligned}
 \text{badago}(x, (z:w) \text{ ++ } r) &= && (\text{++ eragilearen 2. ekuazioa erabiliz}) \\
 = \text{badago}(x, z:(w \text{ ++ } r)) &= && (\underline{\text{badago}} \text{ eragilearen 3. ekuazioagatik}) \\
 = \mathbf{badago}(x, w \text{ ++ } r)
 \end{aligned}$$

$$\begin{aligned}
 \text{badago}(x, z:w) \parallel \text{badago}(x, r) &= && (\underline{\text{badago}} \text{ eragilearen 3. ekuazioagatik}) \\
 = \text{badago}(x, w) \parallel \text{badago}(x, r) &= && (\text{indukzio-hipotesiagatik}) \\
 = \mathbf{badago}(x, w \text{ ++ } r)
 \end{aligned}$$

Ekuazioaren alde bietan emaitza bera lortu da eta ondorioz s zerrenda z:w erakoa denean eta $x \neq z$ denean propietatea bete egiten dela frogatu dugu.

Oinarrizko kasuan s zerrenda hutsa denean ($[]$ erakoa denean) propietatea bete egiten dela frogatu da eta kasu induktiboan s hutsa ez denean ere (hau da, z:w erakoa denean ere) propietatea bete egiten dela frogatu da (bai $x = z$ denean eta baita $x \neq z$ denean ere). Beraz propietatea beti betetzen dela frogatu dugu, x edozein balio eta s eta r edozein bi zerrenda izanda.

4. adibidea

Edozein hiru zerrenda hartuta (s, r eta q) honako hau betetzen dela frogatuko dugu:

$$(s ++ r) ++ q = s ++ (r ++ q)$$

Propietate honek ++ eragilea elkarkorra dela dio.

Indukzioa s zerrendarekiko burutuko da.

- Oinarrizko kasua: s zerrenda hutsa da, hau da, $s = []$

$$([], ++ r) ++ q = [] ++ (r ++ q)?$$

Ekuazioaren alde biek balio bera dutela egiaztatu behar da.

$$\begin{aligned} ([] ++ r) ++ q &= & (\underline{++} \text{ eragilearen 1. ekuazioagatik}) \\ &= \mathbf{r ++ q} \end{aligned}$$

$$\begin{aligned} [] ++ (r ++ q) &= & (\underline{++} \text{ eragilearen 1. ekuazioagatik}) \\ &= \mathbf{r ++ q} \end{aligned}$$

Ekuazioaren alde bietan emaitza bera lortu da eta ondorioz s zerrenda hutsa denean propietatea bete egiten dela frogatu dugu.

- Kasu induktiboa: s ez da zerrenda hutsa, hau da, $s = x:w$

Honako hau frogatu behar dugu:

$$((x:w) ++ r) ++ q = (x:w) ++ (r ++ q)$$

- **Indukzio-hipotesia** (ih): w, r eta q zerrendek propietatea bete egiten dutela suposatuko dugu

$$(w ++ r) ++ q = w ++ (r ++ q)$$

- **Indukzio-urratsa**: w, r eta q zerrendek propietatea bete egiten dutela suposatuz, $x:w$, r eta q zerrendek ere bete egiten dutela frogatuko da. Horretarako ekuazioaren alde biek balio bera dutela egiaztatu behar da:

$$\begin{aligned} ((x:w) ++ r) ++ q &= & (\underline{++} \text{ eragilearen 2. ekuazioagatik}) \\ &= (x:(w ++ r)) ++ q = & (\underline{++} \text{ eragilearen 2. ekuazioagatik}) \\ &= \mathbf{x:((w ++ r) ++ q)} \end{aligned}$$

$$\begin{aligned} (x:w) ++ (r ++ q) &= & (\underline{++} \text{ eragilearen 2. ekuazioagatik}) \\ &= x:(w ++ (r ++ q)) = & (\text{Indukzio-hipotesiagatik}) \\ &= \mathbf{x:((w ++ r) ++ q)} \end{aligned}$$

Ekuazioaren alde bietan emaitza bera lortu da eta ondorioz s zerrenda hutsa ez denean, hau da, $x:w$ erakoa denean propietatea bete egiten dela frogatu dugu.

Oinarrizko kasuan, s zerrenda hutsa denean ($[]$ erakoa denean) propietatea bete egiten dela frogatu da eta kasu induktiboan, s hutsa ez denean ere (hau da, $x:w$ erakoa denean ere) propietatea bete egiten dela frogatu da. Beraz propietatea beti betetzen dela frogatu dugu, s , r eta q edozein hiru zerrenda izanda.

5. adibidea

Mota berekoak diren eta luzera bera duten edozein bi zerrenda hartuta (r eta s) honako propietate hau betetzen dela frogatu dugu:

$$\text{luzera}(\text{tartekatu}(r, s)) = 2 * \text{luzera}(r)$$

Indukzioa r zerrendarekiko burutuko da.

- Oinarrizko kasua: r zerrenda hutsa da, hau da, $r = []$

$$\text{luzera}(\text{tartekatu}([], s)) = 2 * \text{luzera}([])?$$

Ekuazioaren alde biak kalkulatu behar dira propietatea bete egiten dela egiaztatzeko. Gogoratu r eta s zerrendek luzera bera dutela eta ondorioz r hutsa baldin bada s ere hutsa izango da:

$$\begin{aligned} \text{luzera}(\text{tartekatu}([], s)) &= (\text{tartekatu eragiketaren 1. ekuazioagatik}) \\ &= \text{luzera}([]) = (\text{luzera eragiketaren 1. ekuazioagatik}) \\ &= 0 \end{aligned}$$

$$\begin{aligned} 2 * \text{luzera}([]) &= (\text{luzera eragiketaren 1. ekuazioagatik}) \\ &= 2 * 0 = (0 \text{ biderketaren elementu nulua delako}) \\ &= 0 \end{aligned}$$

Ekuazioaren alde bietan emaitza bera lortu da eta ondorioz r zerrenda hutsa denean propietatea bete egiten dela frogatu dugu.

- Kasu induktiboa: r ez da zerrenda hutsa, hau da, $r = x:w$

Honako hau frogatu behar da:

$$\text{luzera}(\text{tartekatu}(x:w, s)) = 2 * \text{luzera}(x:w)$$

Kasu honetan w eta $\text{hond}(s)$ zerrendentzat propietatea bete egiten dela suposatuz, $x:w$ eta s -rentzat ere bete egiten dela frogatu behar da. Horretarako ekuazioaren alde biak kalkulatu behar dira berdinak direla egiaztatuz.

- **Indukzio-hipotesia** (ih): w eta $\text{hond}(s)$ zerrendek propietatea bete egiten dutela suposatuko dugu. Kasu honetan w eta $\text{hond}(s)$ zerrendek luzera bera izango dute:

$$\text{luzera}(\text{tartekatu}(w, \text{hond}(s))) = 2 * \text{luzera}(w)$$

- **Indukzio-urratsa:** w eta $\text{hond}(s)$ zerrendek propietatea bete egiten dutela suposatuz, $x:w$ eta s zerrendek ere bete egiten dutela frogatu behar da.

$$\text{luzera}(\text{tartekatu}(x:w, s)) = 2 * \text{luzera}(x:w)?$$

Propietatea bete egiten dela frogatzeko ekuazioaren alde biak kalkulatu beharko dira emaitza bera ematen dutela egiaztatuz.

$$\begin{aligned} \text{luzera}(\text{tartekatu}(x:w, s)) &= (\text{tartekatu eragiketaren 2. ekuazioagatik}) \\ &= \text{luzera}(x:\text{leh}(s):\text{tartekatu}(w, \text{hond}(s))) = \\ &\quad (\text{luzera eragiketaren 2. ekuazioagatik}) \\ &= 1 + \text{luzera}(\text{leh}(s):\text{tartekatu}(w, \text{hond}(s))) = \\ &\quad (\text{luzera eragiketaren 2. ekuazioagatik}) \\ &= \underline{1} + (1 + \text{luzera}(\text{tartekatu}(w, \text{hond}(s)))) = \\ &\quad (\text{batuketa elkarkorra delako}) \\ &= 2 + \text{luzera}(\text{tartekatu}(w, \text{hond}(s))) \end{aligned}$$

$$\begin{aligned} 2 * \text{luzera}(x:w) &= (\text{luzera eragiketaren 2. ekuazioagatik}) \\ &= \underline{2} * (1 + \text{luzera}(w)) = (* \text{ banakorra delako batuketarekiko}) \\ &= 2 + (2 * \text{luzera}(w)) = (\text{Indukzio-hipotesiagatik}) \\ &= 2 + \text{luzera}(\text{tartekatu}(w, \text{hond}(s))) \end{aligned}$$

Ikus daitekeenez, ekuazioaren alde biak garatuz gauza bera lortu da eta ondorioz r zerrenda hutsa ez denean, hau da, $x:w$ erakoa denean ere propietatea bete egiten dela frogatu dugu.

Beraz, r zerrenda hutsa denean ($[]$ erakoa denean) propietatea bete egiten dela frogatu da eta r hutsa ez denean ere (hau da, $x:w$ erakoa denean ere) bete egiten dela frogatu da, ondorioz beti betetzen da. Baina propietatea zerrenda biek (r eta s) luzera bera dutenean bakarrik beteko da, izan ere luzera berekoak ez badira *tartekatu* funtzioak errorea itzuliko du. Froga egiterakoan hasieratik zerrenda biak luzera berekoak direla kontsideratu da.

5.3.6. Espresioen sinplifikazioa

Eragiketak definitzen dituzten ekuazioak eta eragiketen propietateak kontuan hartuz espresioak sinplifikatzeko aukera egoten da kasu batzuetan. Jarraian lau adibide erakutsiko dira eta adibide horietan sinplifikazio-urrats bakoitzean zein ekuazio edo zein propietate erabili den zehaztuko da:

1. adibidea

$\text{luzera}(\text{hond}(x:s)) \rightarrow \text{luzera}(s)$ (hond eragiketaren 2. ekuazioagatik)

2. adibidea

$\text{luzera}(\text{hond}(7:90:5:[])) \rightarrow$ (hond eragiketaren 2. ekuazioagatik)
 $\text{luzera}(90:5:[]) \rightarrow$ (luzera eragiketaren 2. ekuazioagatik)
 $1 + \text{luzera}(5:[]) \rightarrow$ (luzera eragiketaren 2. ekuazioagatik)
 $1 + (1 + \text{luzera}([])) \rightarrow$ (batuketa elkarkorra delako)
 $2 + \text{luzera}([]) \rightarrow$ (luzera eragiketaren 1. ekuazioagatik)
 $2 + 0 \rightarrow$ (0 batuketaren elementu neutroa delako)
 2

3. adibidea

$\text{luzera}((x:s) ++ (\text{hond}(y:z:r))) \rightarrow$ (hond eragiketaren 2. ekuazioagatik)
 $\text{luzera}((x:s) ++ (z:r)) \rightarrow$ (++ eragiketaren 2. ekuazioagatik)
 $\text{luzera}(x:(s ++ (z:r))) \rightarrow$ (luzera eragiketaren 2. ekuazioagatik)
 $1 + \text{luzera}(s ++ (z:r)) \rightarrow$ (Aurreko ataleko 2. adibidean ikusitako propietateagatik)
 $1 + (\text{luzera}(s) + \text{luzera}(z:r)) \rightarrow$ (luzera eragiketaren 2. ekuazioagatik)
 $1 + (\text{luzera}(s) + (1 + \text{luzera}(r))) \rightarrow$ (batuketa elkarkorra delako)
 $1 + ((\text{luzera}(s) + 1) + \text{luzera}(r)) \rightarrow$ (batuketa elkarkorra delako)
 $(1 + (\text{luzera}(s) + 1)) + \text{luzera}(r) \rightarrow$ (batuketa trukakorra delako)
 $(1 + (1 + \text{luzera}(s))) + \text{luzera}(r) \rightarrow$ (batuketa elkarkorra delako)
 $(2 + \text{luzera}(s)) + \text{luzera}(r)$

4. adibidea

$\text{hond}(\text{hond}(8:7:\text{hond}(1:2:[]))) \rightarrow$ (hond eragiketaren 2. ekuazioagatik)
 $\text{hond}(\text{hond}(8:7:2:[])) \rightarrow$ (hond eragiketaren 2. ekuazioagatik)
 $\text{hond}(7:2:[]) \rightarrow$ (hond eragiketaren 2. ekuazioagatik)
 $2:[]$

5.4. Pilak

Pila bat honela adierazi ohi da grafikoki:

2
7
35
-8
0

Pilen mota Haskell-en ez dago aurredefinituta.

5.4.1. Eragiketa eraikitzaileak pilentzat

Datu-mota bati dagozkion eragiketa eraikitzaileak mota horretako balioak zein diren adierazteko dira.

Datu-mota berri bat definitzeko bere eragiketa eraikitzaileak zein diren zehaztu behar da.

Pilen mota honela definituko dugu:

```
data Pila t = Phutsa | Pilaratu (t, Pila t)
```

Pilentzat eragiketa eraikitzaileak Phutsa eta Pilaratu izango dira.

Definizio horren bidez t motako pila bat pila hutsa izango dela edo bestela t motako pila baten gainean t motako elementu bat ipiniz lortutako pila bat izango dela adierazten da. Gogoratu t parametroak edozein mota ordezkatzen duela (Int, Bool, eta abar).

Phutsa eta Pilaratu eragiketa eraikitzaileen motak honako hauek dira:

```
Phutsa :: Pila t
Pilaratu :: (t, Pila t) -> Pila t
```

Phutsa eragiketak t motako pila bat sortuko du ezerezetik (hau da, sarrerako daturik erabili gabe). Pilaratu eragiketak, t motako elementu bat eta t motako pila bat emanda, elementua pilaren gainean ipiniz lortzen den t motako pila itzuliko du.

Har dezagun honako pila hau:

2
7
35

Pila hori honela adieraziko genuke Phutsa eta Pilaratu eragiketak erabiliz:

```
Pilaratu(2, Pilaratu(7, Pilaratu(35, Phutsa)))
```

Espresio horrek honako hau adierazten du:

Hasteko pila hutsa sortuko da:

Pila hutsaren gainean 35 zenbakia ipiniko da (hau da, 35 zenbakia pilaratuko da) eta horrela 35 zenbakia duen pila izango dugu:

35

Pila horren gainean 7 zenbakia ipiniko da, honako pila hau lortuz:

7
35

Bukatzeko pila horren gainean 2 zenbakia ipiniko da honako pila hau lortuz:

2
7
35

Pilen datu-mota parametroduna da, t parametroaren arabera boolearrez osatutako pilak edo zenbaki osoz osatutako pilak edo zenbaki osozko zerrendez osatutako pilak, edo zenbaki osozko pilez osatutako pilak eta abar defini baititzakegu. Beraz, edozein motatako pilak defini daitezke baina pila batean elementu denek mota berekoak izan behar dute.

Adibideak:

Honako hau **Pila Bool** motako pila bat da:

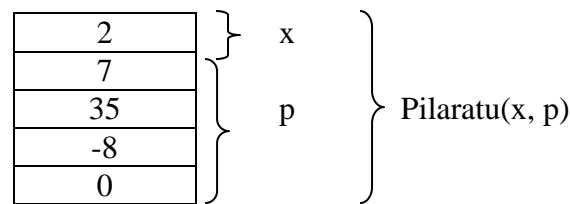
False
True
False
False

Beste hau **Pila [Int]** motakoa da

[0, 5]
[]
[77, -50, 3]

Edozein pila hartuta, pilak honako bi egitura hauetako bat izango du:

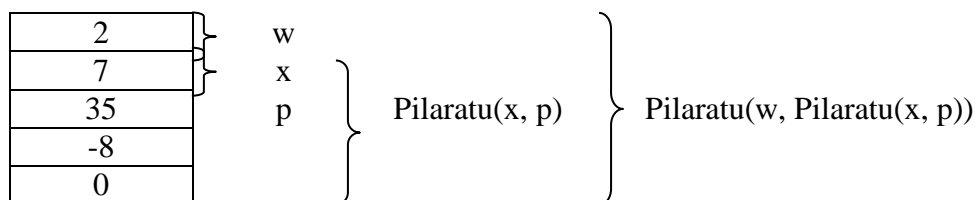
- ✓ Phutsa (pila hutsa denean)
- ✓ Pilaratu(x, p) (x elementua p pila baten gainean ipiniz lortutako pila ez-hutsa da) (x eta p izenak erabili beharrean beste izen batzuk ere erabili daitezke)



Pila batek bi elementu edo gehiago dituenean honako egitura hau duela ere esan daiteke:

$\text{Pilaratu}(w, \text{Pilaratu}(x, p))$

Gailurreko elementua w izango da, gailurretik hasita bigarren elementua x izango da eta gainontzeko elementuek p azpi-pila osatuko dute.



5.4.2. Eragiketa ez-erakitzaile batzuk pilentzat

Jarraian pilekin kalkuluak burutzeko balio duten eragiketa batzuk definituko dira. Funtzio bat edo eragiketa bat definitzeko, funtzioaren mota eta funtzioak zer egiten duen zehazten duten ekuazioak eman behar dira.

- **gailurra**: Pila bat emanda, pilaren gailurrean dagoen elementua itzuliko du funtzio honek. Pila hutsa baldin bada, errore mezu bat aurkeztuko da.

Adibideak:

gailurra(Pilaratu(2, Pilaratu(7, Pilaratu(35, Phutsa)))) = 2
gailurra(Phutsa) = error

Eragiketaren mota:

gailurra:: (Pila t) -> t

Eragiketa definitzen duten ekuazioak:

gailurra(Phutsa) = error "Pila hutsa. Ez dago gailurrik." (1)

gailurra(Pilaratu(x, p)) = x (2)

Errore-mezu bat aurkezteko Haskell-eko *error* funtzioa erabiltzen da eta zein mezu aurkeztu nahi den zehaztu beharko da komatxo bikoitzen artean.

- **despilatu**: Pila bat emanda, pilaren gailurrean dagoen elementua kenduz gelditzen den pila itzuliko du funtzio honek. Pila hutsa baldin bada, errore mezu bat aurkeztuko da.

Adibideak:

```
despilatu(Pilaratu(2, Pilaratu(7, Pilaratu(35, Phutsa)))) =
= Pilaratu(7, Pilaratu(35, Phutsa))
despilatu(Phutsa) = error
```

Eragiketaren mota:

```
despilatu:: (Pila t) -> Pila t
```

Eragiketa definitzen duten ekuazioak:

```
despilatu(Phutsa) = error "Pila hutsa. Ezin da despilatu." (1)
```

```
despilatu(Pilaratu(x, p)) = p (2)
```

Errore-mezu bat aurkezteko Haskell-eko *error* funtzioa erabiltzen da eta zein mezu aurkeztu nahi den zehaztu beharko da komatxo bikoitzen artean.

- **hutsa da**: Pila bat emanda, pila hori hutsa baldin bada, funtzioak True itzuliko du eta bestela False itzuliko du.

Adibideak:

```
hutsa_da(Phutsa) = True
```

```
hutsa_da(Pilaratu(2, Pilaratu(7, Pilaratu(35, Phutsa)))) = False
```

Eragiketaren mota:

```
hutsa_da:: (Pila t) -> Bool
```

Eragiketa definitzen duten ekuazioak:

```
hutsa_da(Phutsa) = True (1)
```

```
hutsa_da(Pilaratu(x, p)) = False (2)
```

Emandako pila hutsa baldin bada, hau da, Phutsa egitura baldin badu, funtzioak True itzuliko du. Emandako pila hutsa ez bada, hau da, Pilaratu(x, p) egitura baldin badu, orduan False itzuliko du.

5.5. Ilarak

Ilaren datu-mota zerrenden antzekoa da baina ilaren kasuan elementu berriak bukaeran ipini behar dira (eskuineko ertzean). Ilarak grafikoki honela adieraziko ditugu:

$$<< 2, 7, 35, -8, 0 >>$$

Ilaren datu-mota ez dago definituta Haskell-en eta Haskell-ek ez du onartzen ilarak $<< 2, 7, 35, -8, 0 >>$ eran adieraztea. Beraz, ilarak adierazteko era hori eragiketen esanahia grafikoki errazago azaltzeko erabiliko dugu baina gero Haskell-en ezingo da erabili.

5.5.1. Eragiketa eraikitzaileak ilara datu-motarentzat

Datu-mota bati dagozkion eragiketa eraikitzaileak mota horretako balioak zein diren adierazteko dira.

Datu-mota berri bat definitzeko bere eragiketa eraikitzaileak zein diren zehaztu behar da.

Ilara mota honela definituko dugu:

$$\text{data Ilara } t = \text{Ihutsa} \mid \text{Ipini}(\text{Ilara } t, t)$$

Ilara datu-motarentzat eragiketa eraikitzaileak Ihutsa eta Ipini dira.

Definizio horren bidez t motako ilara bat ilara hutsa izango dela edo bestela t motako ilara baten bukaeran t motako elementu bat ipiniz lortutako ilara bat izango dela adierazten da. Gogoratu t parametroak edozein mota ordezkatzen duela (Int, Bool, eta abar).

Ihutsa eta Ipini eragiketa eraikitzaileen motak honako hauek dira:

$$\begin{aligned} \text{Ihutsa} &:: \text{Ilara } t \\ \text{Ipini} &:: (\text{Ilara } t, t) \rightarrow \text{Ilara } t \end{aligned}$$

Ihutsa eragiketak t motako ilara bat sortzen du ezerezetik (hau da, sarrerako daturik erabili gabe). Ipini eragiketak, t motako ilara bat eta t motako elementu bat emanda, elementua ilararen bukaeran (eskuineko ertzean) ipiniz lortzen den t motako ilara itzuliko du.

Har dezagun honako ilara hau:

$$<< 35, -8, 0 >>$$

Ilara hori honela adieraziko genuke eragiketa eraikitzaileen bidez:

$$\text{Ipini}(\text{Ipini}(\text{Ipini}(\text{Ihutsa}, 35), -8), 0)$$

Espresio horren bidez ilara hori sortzeko honako urratsak eman direla adierazten da:

Hasteko ilara hutsa sortu:

<< >>

Ilara hutsaren bukaeran 35 zenbakia ipini, 35 zenbakia duen ilara lortuz:

<< 35 >>

Ilara horren bukaeran 7 zenbakia ipini, honako ilara hau lortuz:

<< 35, 7 >>

Azkenik, ilararen bukaeran 2 zenbakia ipini eta honako ilara hau izango dugu:

<< 35, 7, 2 >>

Ilairen datu-mota parametroduna da, t parametroaren arabera boolearrez osatutako ilarak edo zenbaki osoz osatutako ilarak edo zenbaki osozko zerrendez osatutako ilarak, edo zenbaki osozko pilez osatutako ilarak eta abar defini baititzakegu. Beraz, edozein motatako ilarak defini daitezke baina ilara batean elementu denek mota berekoak izan behar dute.

Adibideak:

Honako ilara hau **Ilara Bool** motakoa da

<< True, False, False, False >>

Honako ilara hau **Ilara [Int]** motakoa da

<< [0, 5], [], [77, -50, 3] >>

Edozein ilara hartuta, ilarak honako bi egitura hauetako bat izango du:

- ✓ Ihutsa (ilara hutsa denean)
- ✓ Ipin(c, x) (ilara ez da hutsa, ilarako azken elementua x izango da eta ilarako beste elementu denek c azpi-ilara osatzen dute) (c eta p izenak erabili beharrean beste izen batzuk ere erabil daitezke)

Ilara batek bi elementu edo gehiago dituenean honako egitura hau duela ere esan daiteke:

Ipin(Ipin(c, x), w)

Ilarako azken elementua w da, azken aurrekoa x eta beste elementu denek c azpi-ilara osatzen dute.

5.5.2. Eragiketa ez-eraikitzaile batzuk ilarentzat

Jarraian ilarekin kalkuluak burutzeko balio duten eragiketa batzuk definituko dira. Funtzio bat edo eragiketa bat definitzeko, funtzioaren mota eta funtzioak zer egiten duen zehazten duten ekuazioak eman behar dira.

- **ihutsa da:** Ilara bat emanda, ilara hutsa baldin bada, True itzuliko du eta bestela False itzuliko du.

Adibideak:

ihutsa_da(Ihutsa) = True

ihutsa_da (Ipini(Ipini(Ipini(Ihutsa, 23), 45), 0), 10) = False

Eragiketaren mota:

ihutsa_da:: (Ilara t) -> Bool

Eragiketa definitzen duten ekuazioak:

ihutsa_da(Ihutsa) = True (1)

ihutsa_da(Ipini(c, x)) = False (2)

Emandako ilara hutsa baldin bada, hau da, Ihutsa egitura baldin badu, funtzioak True itzuliko du. Emandako ilara hutsa ez bada, hau da, Ipini(c, x) egitura baldin badu, orduan False itzuliko du.

- **hasiera:** Ilara bat emanda, ilararen hasieran (ezkerreko ertzean) dagoen elementua itzuliko du. Ilara hutsa baldin bada, errore mezu bat aurkeztuko da.

Adibideak:

hasiera(Ipini(Ipini(Ipini(Ihutsa, 3), 8), 34), 10) = 3

hasiera(Ihutsa) = error

Eragiketaren mota:

hasiera:: (Ilara t) -> t

Eragiketa definitzen duten ekuazioak:

hasiera(Ihutsa) = error "Ilara hutsa. Ez dago elementurik." (1)

hasiera(Ipini(c, x))

| ihutsa_da(c) = x (2)

| otherwise = hasiera (c) (3)

Errore-mezu bat aurkezteko Haskell-eko *error* funtzioa erabiltzen da eta zein mezu aurkeztu nahi den zehaztu beharko da komatxo bikoitzen artean.

- **kendu**: Ilara bat emanda, ilarako hasierako (ezkerreko ertzeko) elementua kenduz gelditzen den ilara itzuliko du funtzio honek. Ilara hutsa baldin bada, errore mezua aurkeztuko da.

Adibideak:

```
kendu(Ipini(Ipini(Ipini(Ihutsa, 2), 5), 20)) =
                                     = Ipini(Ipini(Ihutsa, 5), 20)
kendu(Ihutsa) = error
```

Eragiketaren mota:

```
kendu:: (Ilara t) -> Ilara t
```

Eragiketa definitzen duten ekuazioak:

```
kendu(Ihutsa) = error "Ilara hutsa. Ezin da elementurik kendu." (1)
```

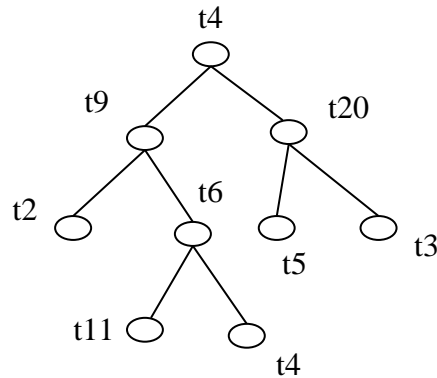
```
kendu(Ipini(c, x))
  | ihutsa_da(c)           = Ihutsa (2)
```

```
  | otherwise             = Ipini(kendu(c), x) (3)
```

Errore-mezu bat aurkezteko Haskell-eko *error* funtzioa erabiltzen da eta zein mezu aurkeztu nahi den zehaztu beharko da komatxo bikoitzen artean.

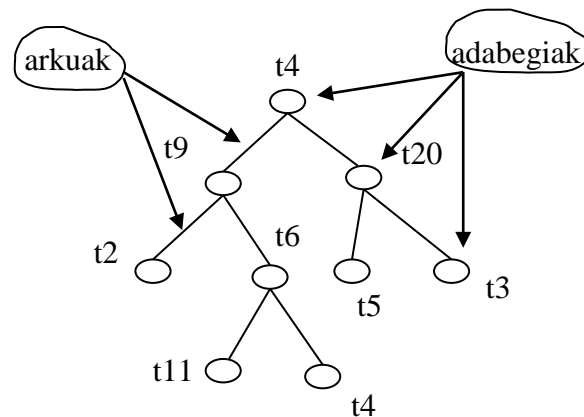
5.6. Zuhaitz bitarrak

t motako zuhaitz bitar bat t motako elementuz osatutako egitura ez-lineal bat da:

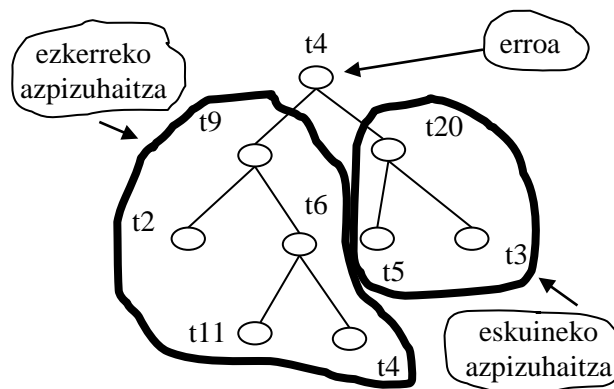


Irudi horretan t2, t3, t4 eta besteak t motako balioak direla suposatuko dugu.

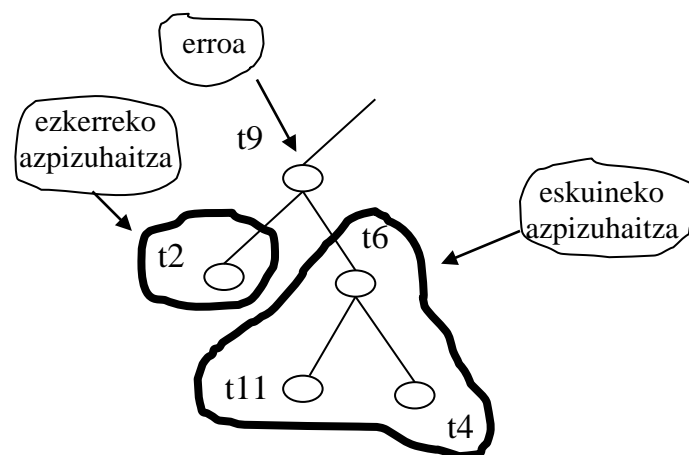
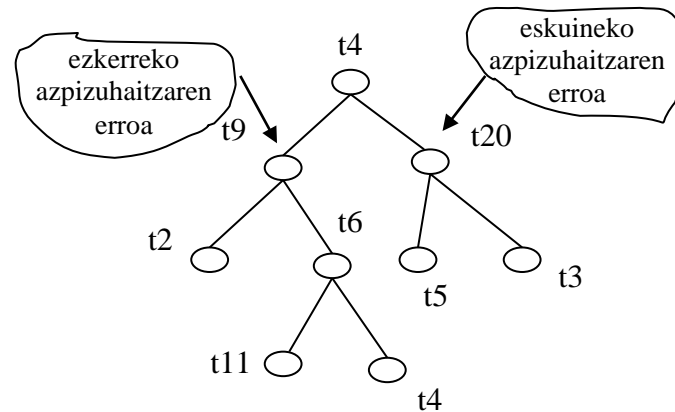
Zuhaitz bitar bat adabegiz eta zuzendutako arkuez (noranzko bakarra duten arkuez) osatuta egoten da. Adabegi bakoitzari t motako balio bat dagokio.



Zuhaitz bitar batean erroa, ezkerreko azpizuhaitza eta eskuineko azpizuhaitza bereiz daitezke:

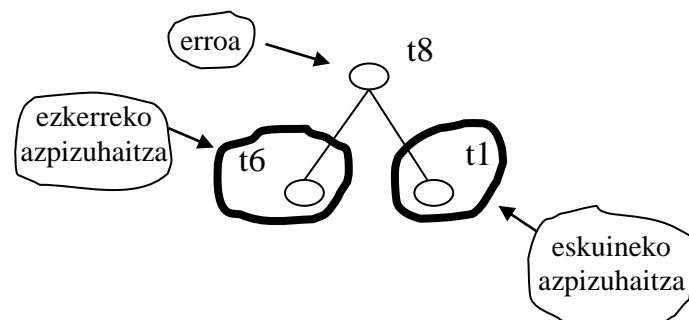


Era berean ezkerreko azpizuhaitzak bere erroa, bere ezkerreko azpizuhaitza eta bere eskuineko azpizuhaitza izango ditu. Eta gauza bera gertatzen da eskuineko azpizuhaitzarekin ere:

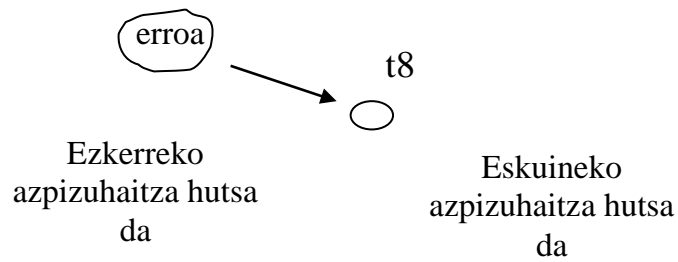


Era horretan zuhaitz batean erroa, ezkerreko azpizuhaitza eta eskuineko azpizuhaitza bereiz daitezke. Eta azpizuhaitzek aldi berean bereizketa bera onartzen dute eta horrela joan gaitezke azpizuhaitz bezala zuhaitz hutsak dituen zuhaitz batera iritsi arte.

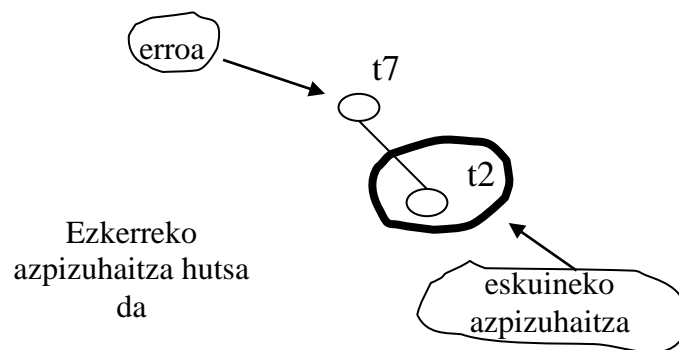
Jarraian ikus daitekeen zuhaitzean bai ezkerreko azpizuhaitza eta bai eskuineko azpizuhaitza adabegi bakar batez osatuta daude:



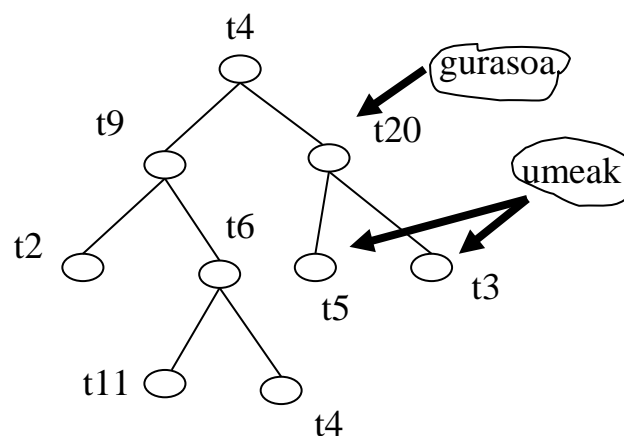
Jarraian datorren zuhaitz honek adabegi bakarra du (erroa) eta ezkerreko eta eskuineko azpizuhaitzak hutsak dira:



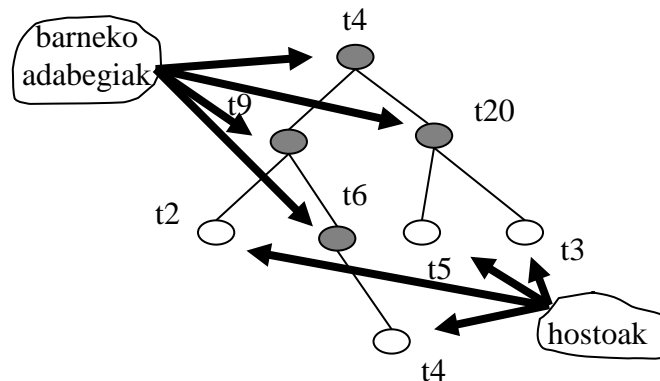
Jarraian ikus daitekeen zuhaitzak bi adabegi ditu. Ezkerreko azpizuhaitza hutsa da:



Adabegi bat bere ezkerreko eta eskuineko azpizuhaitzetako erroen gurasoa dela esaten da. Bi azpizuhaitz horietako erroak ezkerreko umea eta eskuineko umea bezala izendatzen dira, hurrenez hurren. Arkuen noranzkoa gurasoetatik umeenganakoa dela kontsideratzen da:



Azpizuhaitz biak hutsak dituen adabegiak hostoak direla esango dugu. Gutxienez azpizuhaitzetako bat hutsa ez duten adabegiei barneko adabegi deituko diegu.



5.6.1. Eragiketa eraikitzaileak zuhaitz bitarrentzat

Datu-mota bati dagozkion eragiketa eraikitzaileak mota horretako balioak zein diren adierazteko dira.

Zuhaitz bitarren datu-mota ez dago aurredefinituta Haskell-en.

Datu-mota berri bat definitzeko bere eragiketa eraikitzaileak zein diren zehaztu behar da.

Zuhaitz bitarren datu-mota honela definituko dugu:

```
data Zuhbit t = Zhutsa | Eraiki(t, Zhutsa t, Zhutsa t)
```

Zuhaitz bitarrentzat eragiketa eraikitzaileak Zhutsa eta Eraiki dira.

Definizio horren bidez t motako zuhaitz bitar bat zuhaitz bitar hutsa izango dela (hori Zhutsa-ren bidez adieraziko da) edo bestela t motako elementu bat eta t motako bi zuhaitz bitar elkartuz eraikitako zuhaitz bat izango dela adierazten da. Elementu hori zuhaitz berriaren erroa izango da, lehenengo zuhaitza zuhaitz berriaren ezkerreko azpizuhaitza izango da eta bigarren zuhaitza zuhaitz berriaren eskuineko azpizuhaitza izango da. Gogoratu t parametroak edozein mota ordezkatzeko duela (Int, Bool, eta abar).

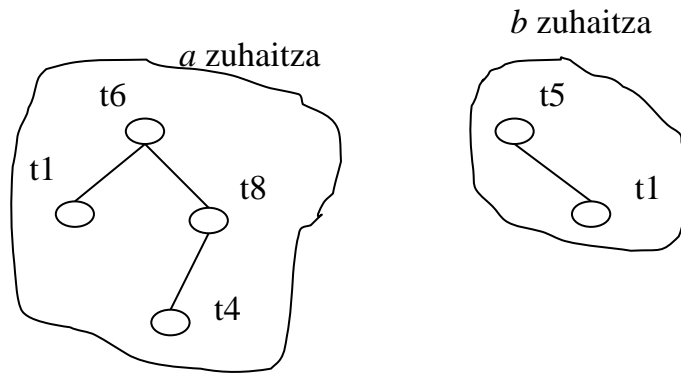
Beste era batera esanda, eragiketa eraikitzaileak ezerezetik (hau da, sarrerako daturik gabe) zuhaitz hutsa sortzen duen eragiketa eta t motako balio bat eta bi zuhaitz emanda, zuhaitz berria eraikitzen duen eragiketak dira.

Zhutsa eta Eraiki eragiketa eraikitzaileen motak honako hauek dira:

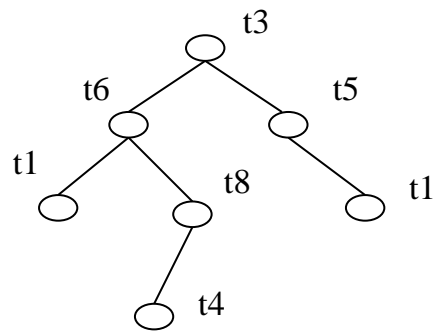
```
Zhutsa :: Zuhbit t
Eraiki :: (t, Zuhbit t, Zuhbit t) -> Zuhbit t
```

.

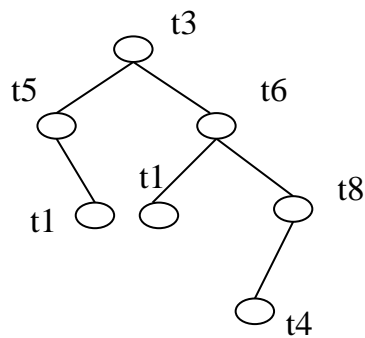
Demagun honako a eta b zuhaitz bitarrak ditugula:



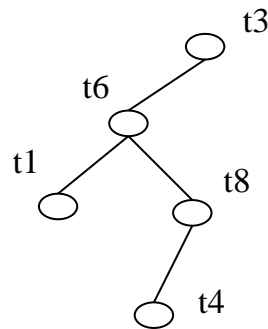
Eraiki($t3$, a , b) zuhaitza honako zuhaitz hau da:



Eraiki($t3$, b , a) zuhaitza honako zuhaitz hau da:



Eraiki($t3, a, \text{Zhutsa}$) zuhaitza honako zuhaitz hau da:



Eraiki($t3, \text{Zhutsa}, \text{Zhutsa}$) zuhaitza honako zuhaitz hau da:

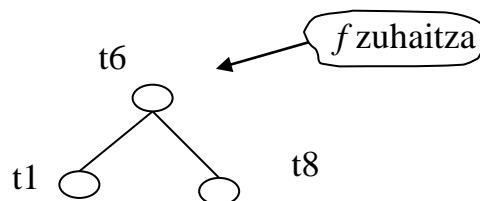


Edozein zuhaitz bitar eragiketa eraikitzaileen bidez adieraz daiteke.

Zuhaitz hutsa honela adieraziko genuke:

Zhutsa

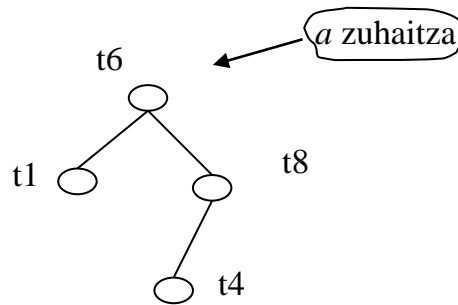
Har dezagun orain f bezala izendatuko dugun honako zuhaitz bitar hau:



f zuhaitz bitarra honela adieraziko genuke:

$f = \text{Eraiki}(t6, \text{Eraiki}(t1, \text{Zhutsa}, \text{Zhutsa}), \text{Eraiki}(t8, \text{Zhutsa}, \text{Zhutsa}))$

Orain har dezagun beste zuhaitz bitar hau



Eragiketa eraikitzaileen bidez honela adieraziko genuke:

$$a = \text{Eraiki}(t6, \text{Eraiki}(t1, \text{Zhutsa}, \text{Zhutsa}), \text{Eraiki}(t8, \text{Eraiki}(t4, \text{Zhutsa}, \text{Zhutsa}), \text{Zhutsa}), \text{Zhutsa})$$

5.6.2. Eragiketa ez-eraikitzaile batzuk zuhaitz bitarrentzat

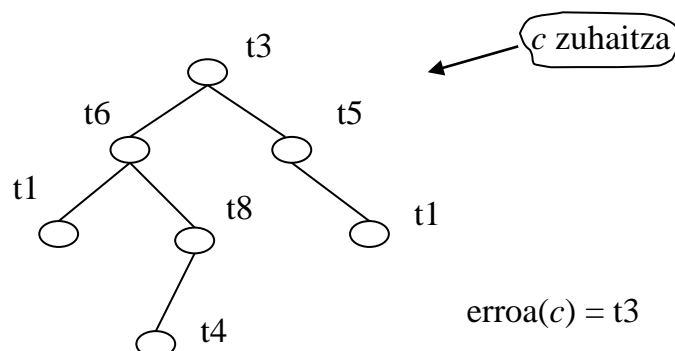
Zuhaitz bitarrekin kalkuluak burutzeko balio duten eragiketa ez eraikitzaile batzuk definituko dira jarraian. Eragiketa edo funtzio bat definitzeko funtzioaren mota eta funtzioa definitzen duten ekuazioak eman behar dira.

- **erroa:** zuhaitz bitar bat emanda, bere erroari dagokion balioa itzuliko du. Zuhaitza hutsa baldin bada errore-mezua aurkeztuko da.

Adibideak:

$\text{erroa}(\text{Zhutsa}) = \text{error}$

Har dezagun honako zuhaitz hau:



$\text{erroa}(c) = t3$

Eragiketaren mota:

$\text{erroa}:: (\text{Zuhbit } t) \rightarrow t$

Eragiketa definitzen duten ekuazioak:

$\text{erroa}(\text{Zhutsa}) = \text{error "Zuhaitz hutsa. Ez dago errorik."}$ (1)

$\text{erroa}(\text{Eraiki}(x, a, b)) = x$ (2)

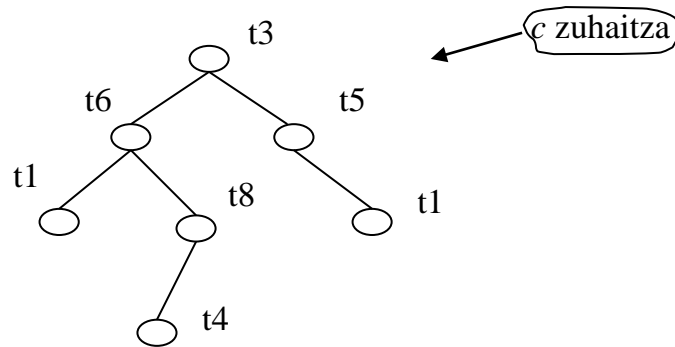
Errore-mezu bat aurkezteko Haskell-eko *error* funtzioa erabiltzen da eta zein mezu aurkeztu nahi den zehaztu beharko da komatxo bikoitzen artean.

- **ezker:** zuhaitz bitar bat emanda, funtzio honek ezkerreko azpizuhaitza itzuliko du. Datu bezala emandako zuhaitza hutsa baldin bada, errore mezua aurkeztu beharko du.

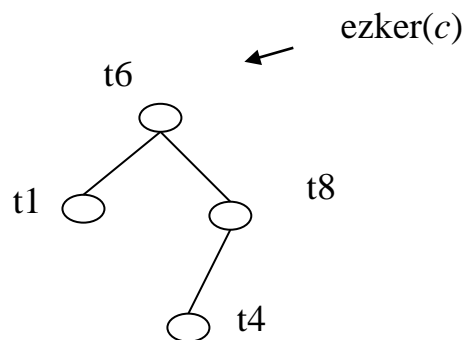
Adibideak:

$\text{ezker}(\text{Zhutsa}) = \text{error}$

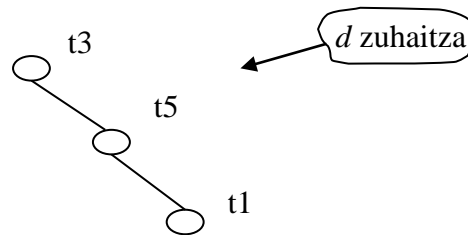
c zuhaitza honako hau baldin bada:



ezker izeneko eragiketak ezkerreko azpizuhaitza itzuliko du:



d honako zuhaitz bitar hau baldin bada:



$$\text{ezker}(d) = \text{Zhutsa}$$

ezker izeneko eragiketak ezkerreko azpizuhaitza itzuliko du. Kasu honetan ezkerreko azpizuhaitza zuhaitz hutsa da:

Eragiketaren mota:

ezker:: (Zuhbit t) -> Zuhbit t

Eragiketa definitzen duten ekuazioak:

ezker(Zhutsa) = error "Zhuhaitz hutsa. Ez dago ezkerreko azpizuhaitzik." (1)

ezker(Eraiki(x , a , b)) = a (2)

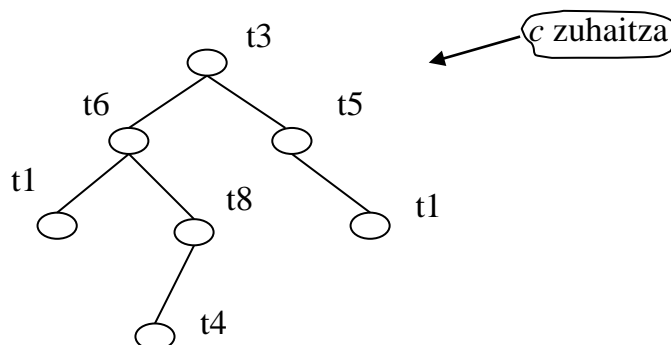
Errore-mezu bat aurkezteko Haskell-eko *error* funtzioa erabiltzen da eta zein mezu aurkeztu nahi den zehaztu beharko da komatxo bikoitzen artean.

- **eskuin**: zuhaitz bitar bat emanda, funtzio honek eskuineko azpizuhaitza itzuliko du. Datu bezala emandako zuhaitza hutsa baldin bada, errore mezua aurkeztu beharko du.

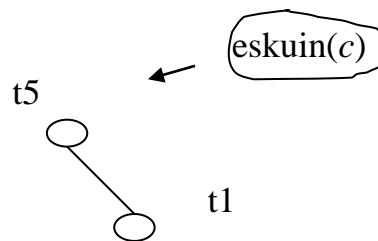
Adibideak:

eskuin(Zhutsa) = error

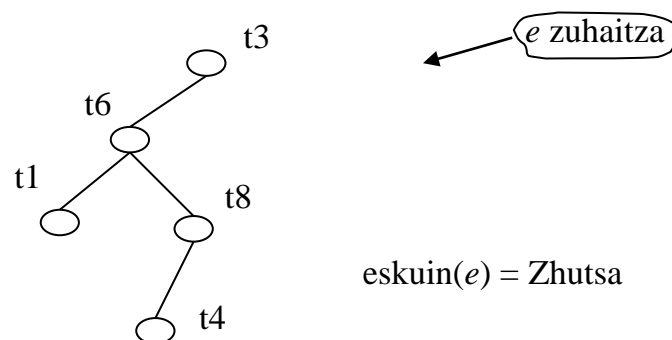
c zuhaitza honako hau baldin bada:



eskuin izeneko eragiketak eskuineko azpizuhaitza itzuliko du:



e zuhaitza honako hau baldin bada:



$\text{eskuin}(e) = \text{Zhutsa}$

Eragiketaren mota:

$\text{eskuin}:: (\text{Zuhbit } t) \rightarrow \text{Zuhbit } t$

Eragiketa definitzen duten ekuazioak:

$\text{eskuin}(\text{Zhutsa}) = \text{error "Zuhaitz hutsa. Ez dago eskuineko azpizuhaitzik."}$ (1)

$\text{eskuin}(\text{Eraiki}(x, a, b)) = b$ (2)

Errore-mezu bat aurkezteko Haskell-eko *error* funtzioa erabiltzen da eta zein mezu aurkeztu nahi den zehaztu beharko da komatxo bikoitzen artean.

- **zhutsa da:** Zuhaitz bitar bat emanda, True itzuliko du zuhaitza hutsa baldin bada eta False itzuliko du hutsa ez bada.

Adibideak:

$\text{zhutsa_da}(\text{Zhutsa}) = \text{True}$

$\text{zhutsa_da}(\text{Eraiki}(2, \text{Eraiki}(7, \text{Eraiki}(6, \text{Zhutsa}, \text{Zhutsa}), \text{Zhutsa}), \text{Zhutsa})) = \text{False}$

Eragiketaren mota:

$\text{zhutsa_da}:: (\text{Zuhbit } t) \rightarrow \text{Bool}$

Eragiketa definitzen duten ekuazioak:

$\text{zhutsa_da}(\text{Zhutsa}) = \text{True}$ (1)

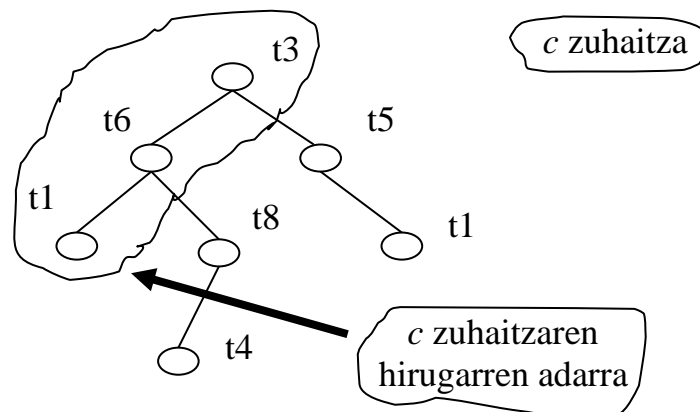
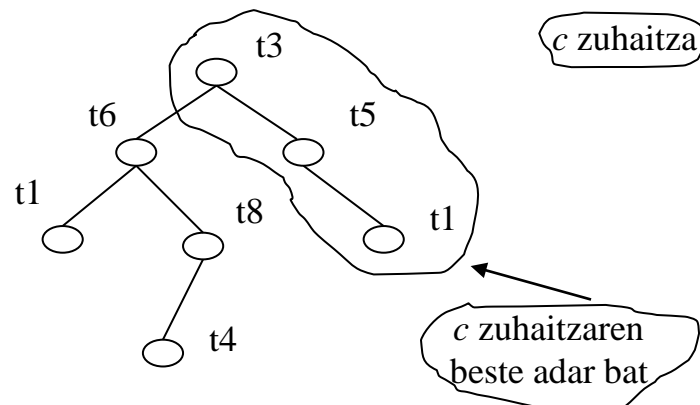
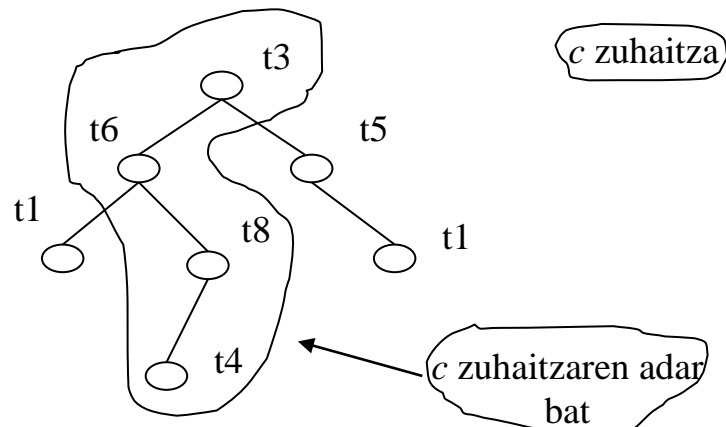
$\text{zhutsa_da}(\text{Eraiki}(x, a, b)) = \text{False}$ (2)

- **sakon:** zuhaitz bitar bat emanda, zuhaitzaren sakonera itzuliko du funtzio honek. Sakonera adar luzeenaren luzera da. Zuhaitza hutsa baldin bada, 0 itzuli beharko du. Adar bat errotik hosto bateraino dauden adabegiei osatzen dute.

Adibideak:

sakon(Zhutsa) = 0

c zuhaitza honako hau baldin bada:



c zuhaitzak hiru adar ditu eta adar luzeenak lau adabegi ditu, beraz:

sakon(c) = 4

Eragiketaren mota:

sakon:: (Zuhaitz t) -> Int

Eragiketa definitzen duten ekuazioak:

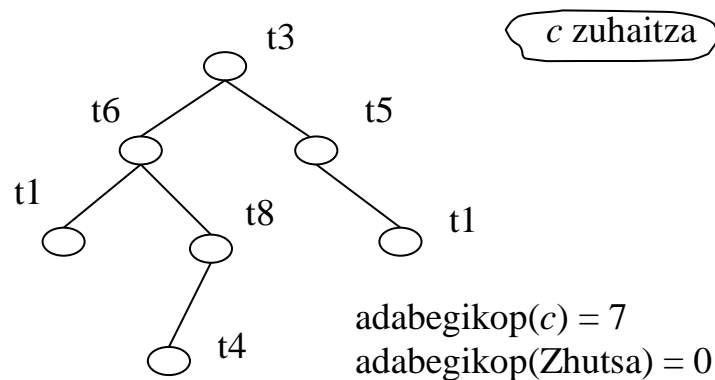
$$\text{sakon}(\text{Zhutsa}) = 0 \quad (1)$$

$$\text{sakon}(\text{Eraiki}(x, a, b)) \quad (2)$$

$$\quad | \text{sakon}(a) \geq \text{sakon}(b) \quad = 1 + \text{sakon}(a)$$

$$\quad | \text{otherwise} \quad = 1 + \text{sakon}(b) \quad (3)$$

- **adabegikop:** Zuhaitz bitar bat emanda, adabegi-kopurua itzuliko du.

Adibideak:**Eragiketaren mota:**

adabegikop:: (Zuhaitz t) -> Int

Eragiketa definitzen duten ekuazioak:

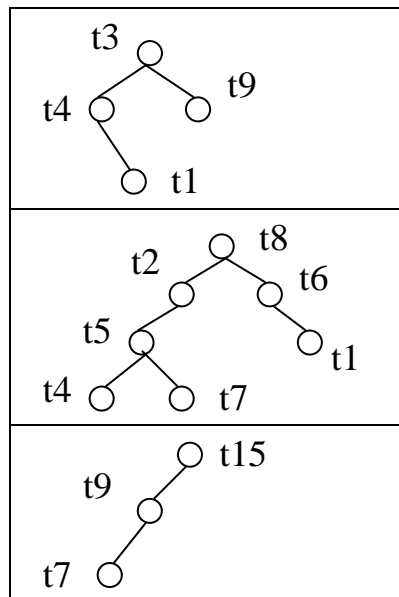
$$\text{adabegikop}(\text{Zhutsa}) = 0 \quad (1)$$

$$\text{adabegikop}(\text{Eraiki}(x, a, b)) = 1 + \text{adabegikop}(a) + \text{adabegikop}(b) \quad (2)$$

5.7. Datu-mota abstraktuak nahasian dituzten adibideak

Zuhaitz bitarrez osatutako pila bat emanda, *sakonh* eragiketak sakonera handiena duen zuhaitzaren sakonera itzuliko du:

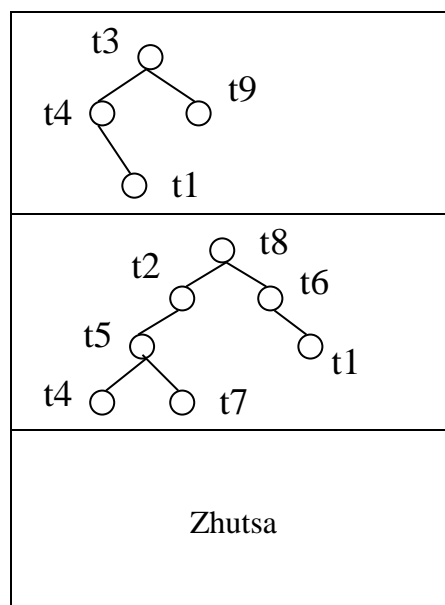
zuhaitz bitarrez osatutako honako pila hau p baldin bada:



Sakonera handiena erdian dagoen zuhaitzari dagokio eta 4 da:

$$\text{sakonh}(p) = 4$$

Horrelako pila batean zuhaitz hutsak ere ager daitezke. Demagun q honako pila hau dela:



$\text{sakonh}(q) = 4$

Eragiketaren mota:

$\text{sakonh}:: \text{Pila (Zuhbit } t) \rightarrow \text{Int}$

Eragiketa definitzen duten ekuazioak:

$\text{sakonh}(\text{Phutsa}) = 0 \quad (1)$

$\text{sakonh}(\text{Pilaratu}(a, p))$

$\text{sakon}(a) \geq \text{sakonh}(p)$	$= \text{sakon}(a)$	(2)
otherwise	$= \text{sakonh}(p)$	(3)

Hor a zuhaitz bitar bat da, p zuhaitz bitarrez osatutako pila bat da eta *sakon* zuhaitz bitar baten sakonera kalkulatzeko funtzioa da.

5.8. Modularitatea Haskell-en

5.8.1. Bi moduluren definizioa aurreko ataletan emandako eragiketak erabiliz

Haskell-en modulu desberdinak definitzea eta modulu batetik beste modulu batean definitutako funtzioak erabiltzea oso erraza da.

Esate baterako, gai honetan definitu ditugun funtzioak kontuan hartuz bi modulu defini ditzakegu:

- a) Modulu bat oinarritzko motentzat definitutako funtzioekin (bikoitia, bakoitia, hand3)
- b) Beste modulu bat zerrendentzat definitutako funtzioekin (leh, hond, hutsa_da, badago, luzera, bikop, tartekatu, tartekatu2)

Modulu bakoitza fitxategi desberdin batean joango da. Orain fitxategi bakoitza nola gelditzen den erakutsiko da.

- a) "Oinarrizkoak.hs" fitxategia

```
module Oinarrizkoak where
```

```
bikoitia:: (Int) -> Bool
```

```
bikoitia (x) = (x `mod` 2) == 0
```

```
bakoitia:: (Int) -> Bool
```

```
bakoitia (x) = not (bikoitia(x))
```

```
hand3:: (Int, Int, Int) -> Int
```

```
hand3 (x, y, z)
```

```
    | x >= y && x >= z           = x
```

```
    | y > x && y >= z           = y
```

```
    | otherwise                 = z
```

b) "Zerrendak.hs" fitxategia

```

module Zerrendak where
import Oinarrizkoak

leh:: ([t]) -> t
leh([]) = error "Zerrenda hutsa. Ez dago lehenengo elementurik."
leh(x:s) = x

hond:: ([t]) -> [t]
hond([]) = error "Zerrenda hutsa. Ez dago hondarrik."
hond(x:s) = s

hutsa_da:: ([t]) -> Bool
hutsa_da([]) = True
hutsa_da(x:s) = False

badago:: (t, [t]) -> Bool
badago(x, []) = False
badago(x, y:s)
    | x == y      = True
    | x /= y      = badago(x, s)

luzera:: ([t]) -> Int
luzera([]) = 0
luzera(x:r) = 1 + luzera(r)

bikop:: ([Int]) -> Int
bikop([]) = 0
bikop(x:r)
    | bikoitia(x)      = 1 + bikop(r)
    | bakoitia(x)      = bikop(r)

tartekatu:: ([t], [t]) -> [t]
tartekatu([], s)
    | not hutsa_da(s)      = error "Luzera desberdineko zerrendak."
    | otherwise            = []
tartekatu(x:r, s)
    | luzera(x:r) /= luzera(s) = error "Luzera desberdineko zerrendak."
    | otherwise            = x:(leh(s): tartekatu(r, hond(s)))

tartekatu2:: ([t], [t]) -> [t]
tartekatu2([], s)
    | not hutsa_da(s)      = error "Luzera desberdineko zerrendak."
    | otherwise            = []
tartekatu2(x:r, s)
    | luzera(x:r) /= luzera(s) = error "Luzera desberdineko zerrendak."
    | otherwise            = leh(s):(x: tartekatu2(r, hond(s)))

```

[] eta : eragile eraikitzaileak eta ++ eragilea aurredefinituta daude Haskell-en.

5.9. Oharrak

5.9.1. *Maiuskula eta minuskulen erabilera izenetan*

Maiuskulak eta minuskulak ondo erabiltzeko honako arau hauek jarraitu behar dira:

- Datu moten izenak maiuskulaz hasten dira: Int, Bool, Char, ...
- Moten ordeztu diren aldagaien izenak minuskulaz hasten dira: t.
- True eta False konstanteak maiuskulaz hasten dira.
- Eraikitzaileak ez diren beste eragile denak eta aldagaien izenak minuskulaz hasten dira (maiuskulak erdian ipintzea egon arren): bikoitia, bakoitia, leh, hond, badago, luzera, x, s, r, p, q, zatb, kenduLuz, kenduLuzBik, ...

5.9.2. *Azalpenak ipintzeko bi era*

Haskell-ez idatzitako programetan azalpenak ipini nahi izanez gero, bi aukera daude:

- -- erabiliz:
-- sinboloen atzetik lerroa bukatu arte datorren dena azalpena izango da.

Adibidea:

--Hau azalpen bat da eta lerroa bukatzen denean bukatuko da azalpena

- {- eta -} erabiliz:
{- eta -} sinboloen artean doan dena azalpena da. Lerro batetik bestera pasatuta ere azalpenak aurrera jarraitzen du.

Adibidea:

{- Hau azalpena da baina
lerroz alda gaitezke. -}