2. PROGRAMEN ESPEZIFIKAZIOA

2.1. Asertzioak: egoera-multzoak adierazteko formulak.

2.2. Aurre-ondoetako espezifikazio formala.

2.1. Asertzioak: egoera-multzoak adierazteko formulak.

Programa baten konputazio-egoera: programa bateko aldagai guztien balioen deskribapena exekuzioaren une jakin batean.

Egoera : {aldagaiak} → {balioak}

Agindu baten exekuzioa = egoera-aldaketa

Konputazioa = egoeren segida

Programa, azken batean, funtzio bat bezala ikus daiteke:

Programa : hasierako egoera → bukaerako egoera

Asertzioak.

- espresio logikoak dira programetan txertatzen direnak iruzkinen modura ({....})
- programako aldagaiek puntu jakin batean betetzen dituzten propietateak adierazten dituzte

  Asertzio bakoitzak puntu batean gerta daitezkeen egoeren multzoa (infinitua izan daitekeena) errepresentatzen du: "egiazko egiten duten egoeren multzoa"

- Adibidea: {z = x+y} asertzioak infinitu egoera errepresentatzen ditu
  - egiazkoa da egoera honetan {x=3 , y=2 , z=5}
  - faltsua da egoera honetan {x=1 , y=2 , z=5}
  - zehaztugabea da egoera honetan {x=1 , y=2}

Asertzioak (adibidea).

Espezifikazioa eta asertzioak iruzkinen bidez

A: array (1..n) of Integer;

```
begin                          { n>=1 } = Aurre
i:=0; b:=0;
while i<n loop                 {  {b = Σ_{j=1}^{i} A(j)} }
     i:=i+1;
     b:=b+A(i);
end loop;
end                            { {b = Σ_{j=1}^{n} A(j)} } = Post
```

Asertzioak idazteko lengoaia: Lehen mailako logika.

- Alfabetoa: ADAri egokitua
  aldagai-sinboloak (x, y, z, ...)
  konstante-sinboloak (0, 1,...a, b, c, ...)
  funtzio-sinboloak (+, -, mod,...,Σ, Π, ...)
  predikatu-sinboloak (<, =, >,..., bikoiti, bakoiti, ...)
- Sintaxia:
  - **Terminoak** dira:
    x aldagaia
    k konstantea
    $f(t_1,..,t_n)$, non f funtzio n-tarra eta $t_1,..,t_n$ terminoak diren.

Asertzioak idazteko lengoaia: Lehen mailako logika (II)

- **Formulak** :
  - **atomoak**:

    p(t1,..,tn), non p predikatu n-tarra eta t_1,..,t_n terminoak diren.

    True, False
  - **konposatuak:**

    eragile logikoen eta kuantifikatzaileen bidez:$\neg\varphi$, $\varphi\wedge\psi$, $\varphi\vee\psi$, $\varphi\rightarrow\psi$, $\varphi\leftrightarrow\psi$, $\exists x\varphi$, $\forall x\varphi$

Asertzioak idazteko lengoaia: Lehen mailako logika (III)

- Semantika:

Formula bakoitzari balio bat (egiazkoa, faltsua) esleitzen die, konputazio-egoeran aldagaiek dituzten balioen arabera.

- Aldagai libre eta atxikiak:

  Aldagai bat *librea* da formula batean ez badago inongo kuantifikatzaileren eraginpean.
  Libreak ez diren aldagaiei *atxiki* deituko diegu.

"asertzio batek EZ du zentzurik baldin eta programan agertzen ez diren aldagai libreak baditu"

Asertzioak idazteko lengoaia: Lehen mailako logika (IV)

- Adibidea:

x=3, y=4 konputazio-egoeran :

$x \geq y$ **faltsua** da

$x+1=y \wedge y>2$ **egiazkoa** da

$(x \bmod 2=0) \leftrightarrow (y \bmod 2=0)$ **faltsua** da

$\exists z(z>0 \wedge z+x=y)$ **egiazkoa** da

$\forall z(1<z \rightarrow z+x>y)$ **egiazkoa** da

Asertzioak idazteko lengoaia: Lehen mailako logika (V)

- Aldagai atxikiak: $\exists x(\varphi)$ eta $\forall x(\varphi)$ kuantifikatzaileen formuletan agertzen diren x aldagaiak.
- Aldagai libreak: Kuantifikatzaileei loturik ez dauden aldagaiak
- Egoera konkretu batean, formula batek balioa duela esango dugu (definitua dagoela), baldin eta soilik baldin bere aldagai libre guztiek balioren bat badute egoera horretan.
- Aldagai atxikiak aldagai laguntzaileak dira nolabait:
  $\exists x(\varphi)$ = "Existitzen da elementu bat (dei diezaiogun x), $\varphi(x)$ betetzen duena"
  $\forall x(\varphi)$ = "Edozein elementuk (dei diezaiogun x), $\varphi(x)$ betetzen du"

Asertzioak idazteko lengoaia: Lehen mailako logika (VI)

- Asertzio batek aldagai libreak baldin baditu eta aldagai libre horiek ez badira programan agertzen, asertzio horrek ez du zentzurik izango.
  - *Zehaztugabea delako*
  - *Programaren egoerarik ez duelako definitzen*
- Asertzio batek aldagai atxikiak baldin baditu eta aldagai atxiki horiek programan agertzen badira, asertzio horrek ez du zentzurik izango.
  - *Programan agertzen dena, aldagai libre gisa agertu behar litzatekeelako asertzioan*
- Laburtuz, asertzioek izan ditzakete:

⌦ programakoak diren aldagai libreak

⌦ programakoak ez diren aldagai atxikiak (laguntzaileak).

Asertzioak idazteko lengoaia: Lehen mailako logika (VII)

•**Formula bat ondo definituta** dago egoera batean b.s.b. bere aldagai libre guztiek balio bat badute egoera horretan.

Laburbilduz

Programetako aldagaiak asertzioetan libre agertuko dira eta ez ditugu aldagai atxiki gisa erabiliko, honetarako programan agertzen ez diren aldagai izenak asmatuko ditugu.

Asertzioak idazteko lengoaia: Lehen mailako logika (VII)

•**Formula bat ondo definituta** dago egoera batean b.s.b. bere aldagai libre guztiek balio bat badute egoera horretan.

Laburbilduz

Programetako aldagaiak asertzioetan libre agertuko dira eta ez ditugu aldagai atxiki gisa erabiliko, honetarako programan agertzen ez diren aldagai izenak asmatuko ditugu.

Kuantifikatzaileak eta aldagaiak. Ordezpenak.

Kuantifikatzailea: laburdura bat da, zenbait eragiketa berdin biltzeko erabiltzen dena.

Kuantifikatzailea	$\longleftrightarrow$	erag. atxikitua
$\forall$		$\wedge$
$\exists$		$\vee$

Patroiak:

$\exists z$ (z-ren domeinua $\wedge$ z-ren propietateak)

$\forall z$ (z-ren domeinua $\rightarrow$ z-ren propietateak)

Kuantifikatzaileak (II)

- Kuantifikatzaileak dira ondoko funtzio-sinboloak:

Σ	+
Π	*
$\aleph$	kontaketa (multzoen kardinala)

- Kuantifikatzaileak aldagai (atxiki) bati lotzen dira. Aldagai horrek domeinu baten erreferentzia (esplizitua edo inplizitua) egiten du.

Kuantifikatzaileak (III)

∃ domeinu tarte batez

$\exists i\ (1 \le i \le n \wedge A(i)<0) \equiv (A(1)<0 \vee A(2)<0 \vee .... \vee A(n)<0\ )$

aldagai atxikia (laguntzailea) i da, eta *domeinua* [1..n] da

$\exists k(k \ge 0 \wedge x = 2^k) \equiv (x = 2^0 \vee x = 2^1 \vee .... \vee x = 2^n \vee ....)$

aldagai atxikia (laguntzailea) k da, eta *domeinua* Nat da

- Oro har: $\exists x\ (D(x)$ [$\wedge$] $))$

  "badago D domeinuko elementu bat P betetzen duena"

- D hutsa bada formularen balioa [faltsua] da
  - "ez dago inolako egoerarik egiazkoa egingo duenik"
  - Adibidez: $\exists z\ (1 \le z \le 0 \wedge x = 2^z)$

Kuantifikatzaileak (IV)

∀ domeinu tarte batez

$\forall j(1<j\leq 9 \rightarrow A(j)>0) \equiv (A(2)>0 \wedge A(3)>0 \wedge \ldots \wedge A(9)>0)$

aldagai atxikia (laguntzailea) j da, eta *domeinua* [2..9] da

- Oro har: $\forall x\,(D(x) \rightarrow P(x))$

  "D domeinuko elementu guztiek P betetzen dute"

- D hutsa bada emaitza egiazkoa da
  - "egoera guztiek egiazko egiten dute"
  - Adibidez $\forall j\,(1<j\leq 1 \rightarrow A(j)>0)$

Kuantifikatzaileak (V)

Beste adibide batzuk:

A: array (1..10) of Integer

- "A(1..10) *zenbaki osoko positiboen bektorea da*"

  $\forall j(1 \leq j \leq 10 \rightarrow A(j)>0)$

- "A-*k zehatz-mehatz zenbaki negatibo bat du*"

  $\exists i\ (1 \leq i \leq 10 \wedge A(i)<0 \wedge \forall j(\forall j((1 \leq j \leq 10 \wedge i \neq j) \rightarrow A(j) \geq 0$

- "*Badago zenbaki bat* 5*en zatigarria dena*" $\exists x(x \bmod 5 = 0)$

  Domeinu inplizitua: integer

- "*Edozein zenbaki bikoiti zenbaki ez-lehena da*"

  $\forall x(\text{bikoiti}(x) \rightarrow \neg \text{lehena}(x))$

  non $\text{bikoiti}(x) \equiv (x \bmod 2 = 0)$

  $\text{lehena}(x) \equiv (x>1 \wedge \forall y(2 \leq y<x \rightarrow x \bmod y \neq 0))$

Kuantifikatzaileak (VI)

Funtzio-sinboloak: Σ, Π eta N

$$\sum_{i=1}^{5} A(i) = A(1)+A(2)+A(3)+A(4)+A(5)$$

ald. atxikia i da, domeinua [1..5] da

$$\sum_{i=1}^{0} A(i) = 0$$ domeinu hutsa, 0 (+ neutroa)

$$\prod_{i=4}^{7} f(i) = f(4)*f(5)*f(6)*f(7)$$

ald. atxikia i da, domeinua [4..7] da

$$\prod_{i=4}^{2} f(i) = 1$$ domeinu hutsa, 1 (* neutroa)

Kuantifikatzaileak (VII)

- $\mathcal{N}x(D(x) \wedge P(x))$ ald. atxikia x da, domeinua D da
  - $\mathcal{N}x(1 \leq x \leq 7 \wedge \text{bikoiti}(x)) = 3$ ({2,4,6} multzoaren kardinala)
  - $\mathcal{N}x(1 \leq x \leq 0 \wedge \text{bikoiti}(x)) = 0$ (domeinu hutsa : 0 =kard ϕ)

Kuantifikatzaileak (VIII)

Adibideak: A: array (1..10) of integer

- "*Ako elementu guztien batura 50 baino handiagoa da*"

$$\sum_{i=1}^{10} A(i) > 50$$

- "*Ako k lehenengo elementuen biderkadura da x*"

$$\prod_{i=1}^{k} A(i) = x$$

- "*3 elementua 5 aldiz agertzen da An*"

$$\aleph i(1 \leq i \leq 10 \wedge A(i)=3) = 5$$

- "*Ako elementu errepikatuen kopurua n da*"

$$\aleph x(\text{erre}(x,A)) = n \quad \text{,non}$$

$$\text{erre}(x,A) = \exists i,j\ (1 \leq i \leq 10 \wedge 1 \leq j \leq 10 \wedge i \neq j \wedge A(i)=A(j)=x)$$

Ordezpenak

- Formula bateko aldagaien **ordez** terminoak jartzea erabilgarria da asignazioari buruz arrazoiketak egiteko.

- Izan bitez φ formula, x aldagaia eta t terminoa, φ_x^t adierazpenak φ-ko x-ren agerpen libre guztien ordez t terminoa jarrita lortzen den formula errepresentatzen du.

  (t terminoak ez du eduki behar φ-n kuantifikatuta ageri den aldagairik).

Ordezpenak

Adibideak:

$$(\,\mathrm{x}^3 \geq 0 \wedge x + z < 5\,)_x^{A(i)} = (A(i)^3 \geq 0 \wedge A(i) + z < 5\,)$$

$$(\prod_{i=1}^{k} A(i) = x)_k^8 = (\prod_{i=1}^{8} A(i) = x)$$

$$(\,\exists \mathrm{n}\,(\,\mathrm{bikoiti}(n) \wedge \mathrm{n} > x))_x^{x+y} = \exists \mathrm{n}\,(\,\mathrm{bikoiti}(n) \wedge \mathrm{n} > x + y)$$

Ariketa: $(\,\forall \mathrm{i}\,(1 \leq i \leq \mathrm{n} \rightarrow A(i) = x)_x^{\mathrm{i-1}}$?

Formula ahulago eta gogorragoak

- φ ψ baino gogorragoa da (edo ψ φ baino ahulagoa) bsb

$$\{s \mid s(\varphi)=\text{True}\} \subseteq \{s \mid s(\psi)=\text{True}\}$$

- Adibidez:

$\{s \mid s(\text{lehena}(x) \wedge x > 3)=\text{True}\}$ (x: Natural dela)

$= \{\{ x=5, ...\}, \{ x=7, ...\}, \{ x=11, ...\}, \ldots\}$

$\mid\bigcap$

$\{\{ x=1, ...\}, \{ x=2, ...\}, \{x = 4, ...\}, \{ x=5, ...\}, \{ x=7, ...\}, \ldots\}$

$= \{s \mid s(\neg (x \bmod 3 = 0))=\text{True}\}$

- $s(\varphi \rightarrow \psi)$ = True da edozein s egoeratarako bsb φ gogorragoa baldin bada ψ baino.

Formula ahulago eta gogorragoak. Adibideak

- $x>0$ gogorragoa da $x \geq 0$ baino $(\varphi \rightarrow (\varphi \vee \psi))$ ← $(x>0 \vee x=0)$
- $(x=0 \wedge y=1)$ gogorragoa da $y = z^x$ baino
- $k=0$
- $k=1 \wedge A(1)=0$

} gogorragoak dira $\forall j(1 \leq j \leq k \rightarrow A(j)=0)$ baino

- $\text{lehena}(2) \rightarrow \exists x\ (\text{lehena}(x))$
- $(5 \leq n \wedge A(5)>0) \rightarrow \exists j(1 \leq j \leq n \wedge A(j)>0)$
- $(A(i)>0 \wedge 1 \leq i \leq n) \rightarrow \exists j(1 \leq j \leq n \wedge A(j)>0)$

} $\varphi_x^t \rightarrow \exists x \varphi$

Formula ahulago eta gogorragoak. Adibideak (II)

- $\forall x\ (x^0 = 1) \rightarrow 2^0 = 1$
- $\forall i(1 \leq i \leq n \rightarrow \text{bikoiti}(A(i))) \rightarrow (7 \leq n \rightarrow \text{bikoiti}(A(7)))$
- $\forall j(1 \leq j \leq n \rightarrow A(j) > 0) \rightarrow (1 \leq i \leq n \rightarrow A(i) > 0)$

$\forall x \varphi \rightarrow \varphi_x^t$

- True da formularik ahulena

  {s| s(True)= True} = multzo osoa

- False da formularik gogorrena:

  {s| s(False)= True} = multzo hutsa

2.2.- Aurre-ondoetako espezifikazio formala

Espezifikazioa = zer egiten du programa batek, ez nola egiten duen.

Propietateak:

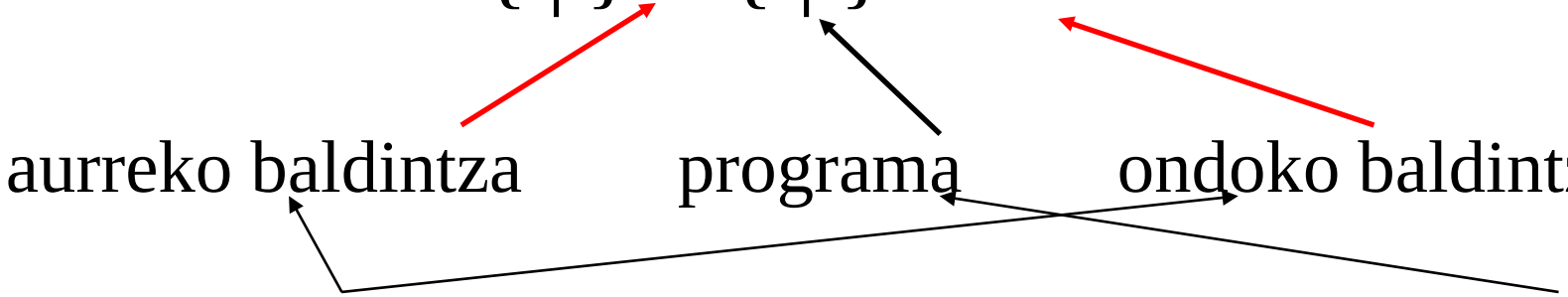
- argitasuna (ulerterraza)
- laburtasuna (erredundantziarik gabea)
- doitasuna (anbiguotasunik ez)
- murriztasun eta orokortasunaren arteko oreka

Espezifikazio-teknika:

formatua + espezifikazio-lengoaia

Aurre-ondoetako espezifikazio formala. Formatua

$$\{\varphi\}\ P\ \{\psi\}$$



asertzioak logika-lengoaian programazio-lengoaia.

Esanahia:

φ betetzen den egoera batean hasten bada P, orduan

ψ betetzen den egoera batean bukatzen da.

(*asertzio bat bete* = asertzioaren balioa True izatea)

- Aurre φ: *datuen baldintza minimoak* (***ahulena***).
- Post ψ: *datu eta emaitzen arteko erlazioa* (***gogorrena***).

Aurre-ondoetako espezifikazio formala.

Adibidea

x eta y osokoen arteko zatidura eta hondar osoa

```
begin          {y>0 ∧ x≥0} = Aurre
r:=x; q:=0;
while r≥y loop
      r:=r-y;
      q:=q+1;
end loop;
end            x = q*y+r ∧ y>r≥0 ∧ q≥0 ∧ q≥0}= Post
```

y>1 ∧ x≥0 gogorregi *Aurre* gisa

- ¬ (y=0) ∧ x≥0 ahulegi *Aurre* gisa
- x = q*y+r ahulegi *Post* gisa
- x = q*y+r ∧ r = y-1 gogorregi *Post* gisa