

3. gaia: Itzultzaile sinple baten definizioa eta garapena (hurbilpena)

3.1 Sarrera

3.2 Lengoaia baten definizio sintaktikoa

3.3 Sintaxiak zuzendutako itzulpena

3.4 Analisi sintaktikoa

3.5 PLen eraikitzaile nagusien itzulpena

3.6 Analisi semantikoa

3.7 Sinbolo-taula

3.8 Bitarteko kode motak

3.9 PLen eraikitzaile nagusien itzulpen hedatua

3.2 Definizio sintaktikoa

Testuingururik gabeko gramatikak

1. Bukaerako ikurren multzoa (tokenak)
2. Ez-bukaerako ikurren multzoa
3. Produkzio-erregelen multzoa. Produkzio bakoitzak ez-bukaerako bat eta bukaerako eta ez-bukaerako ikurren sekuentzia (sekuentzia hutsa izan daiteke)
4. Hasierako ikurra (ez-bukaerakoa)

Testuingururik gabeko gramatikak (II)

- $G = (\Sigma, N, P, S)$
- $P = \{A \rightarrow \alpha : A \in N \wedge \alpha \in (\Sigma \cup N)^*\}$
- Eratorpena:
 $\alpha_1 \Rightarrow \alpha_2$ bsb $\alpha_1 = \beta_1 A \beta_2$ eta $\alpha_2 = \beta_1 \alpha \beta_2$ non $A \in N$ eta $(A \rightarrow \alpha) \in P$
- Sortutako lengoaia:
 $x \in \Sigma^*$ non $S \xRightarrow{*} x$

Adibidea

$G = (\{+, -, *, /, \text{id}, \text{osoko}\}, \{E\}, P, E)$

$P = \{E \rightarrow E + E, E \rightarrow E - E,$

$E \rightarrow E * E, E \rightarrow E / E,$

$E \rightarrow \text{id}, E \rightarrow \text{osoko}\}$

Adibidea (notazio laburtua)

$E \rightarrow E + E$

| $E - E$

| $E * E$

| E / E

| **id**

| **osoko**

Analisi-zuhaitzak (eratorpen-zuhaitzak)

1. Erroa hasierako ikurrarekin etiketatuta dago.
2. Hosto bakoitza bukaerako ikur batekin edo kate hutsarekin etiketatuta.
3. Barruko nodo bakoitza ez-bukaerako ikur batekin etiketatuta.
4. Ez-bukaerako A ikur bat barruko nodo baten etiketa bada, eta X_1, X_2, \dots, X_n nodo horren semeen etiketak badira, orduan $A \rightarrow X_1 X_2 \dots X_n$ produkzioa da.

Kasu berezi bezala, $A \rightarrow \xi$ badago, orduan A rekin etiketatuko nodoak ξ etiketa duen seme bakarra du.

Anbiguotasuna

- Gramatika bat anbigua izango da gramatikak sortutako kate bati bi analisi-zuhaitz desberdin dagozkionean

Anbiguotasun adibidea

$E \rightarrow E + E$

| $E - E$

| $E * E$

| E / E

| **id**

| **osoko**

Gramatika baliokideak

- Bi gramatika baliokideak dira lengoaia bera definitzen badute
- Adibidez, Gren baliokidea baina anbigua ez dena:

$$\begin{array}{l} E \rightarrow E + T \\ | \quad E - T \\ | \quad T \end{array}$$
$$\begin{array}{l} T \rightarrow T * F \\ | \quad T / F \\ | \quad F \\ F \rightarrow \text{id} \\ | \quad \text{osoko} \end{array}$$

3.3 Sintaxiak zuzendutako itzulpena

- Sintaxiak zuzendutako itzulpen-eskema
- Analisi-zuhaitzaren sakonerako korritzea
- Atributuak

Sintaxiak zuzendutako itzulpen-eskema (SZIE)

- Testuingururik gabeko gramatikaren gainean eraikitzen da
- Osagai sintaktiko bakoitzari atributuak lotzen zaizkio
 - » Bukaerakoak: balioa emana dute (ad. analisi lexikoa)
 - » Ez-bukaerakoak: erregela semantikoen bidez kalkulaturako balioak
- Produkzio bakoitzari lotutako erregela semantikoak
 - » itzulpen prozesua espezifikatzen dute (atributuei balioak emanez)

SZIE interpretatzeko

- Analisi-zuhaitz *dekoratua* eraiki
- **n** adabegia gramatikako **X** ikurrarekin etiketatuta badago, **X.a** idatziko dugu adabegi horretan **X**ren **a** atributuaren balioa adierazteko
- **X.a**-ren balioa **n** adabegian, dagokion erregela semantikoak erabiliaz kalkulatu da

SZIE interpretatzeko (II)

- Analisi zuhaitz bat ebaluatzeko, lehenbizi analisi-zuhaitz dekoratua eraiki, atributu guztiak balio gabe dituela
- Ekintza semantikoak bukaerako beste ikurrak bezala gehitzen zaizkio zuhaitzari
- Ekintza bat egikaritu behar den momentua produkzioaren eskuin partean duen lekuak ematen du

Sakonerako korritzea

- Zuhaitz baten sakonerako korritzea erroan hasi eta hostoetan bukatzen da. Prozesu hau zuhaitzaren azpizuhaitz bakoitzeko errepikatzen da.
- **Ezkerretik eskuinerako** ordena erabiliko da.

```
procedure bisitatu(n: nodoa);  
begin  
  for n-ren m seme bakoitzeko, ezkerretik eskuinera  
    if m nodoa da then bisitatu(m)  
    elsif m erregela semantikoa da then ebaluatu(m)  
    end if  
  end loop  
  n nodoaren erregela semantikoak ebaluatu  
end
```


SZIEaren adibidea (I)

$E \rightarrow E + E$	$\{E.\text{balio} := E_1.\text{balio} + E_2.\text{balio}\}$	(1)
$E - E$	$\{E.\text{balio} := E_1.\text{balio} - E_2.\text{balio}\}$	(2)
$E * E$	$\{E.\text{balio} := E_1.\text{balio} * E_2.\text{balio}\}$	(3)
E / E	$\{E.\text{balio} := E_1.\text{balio} / E_2.\text{balio}\}$	(4)
osoko	$\{E.\text{balio} := \mathbf{osoko}.\text{balio}\}$	(5)

SZIEaren adibidea (II)

$E \rightarrow E + T \quad \{E.\text{balio} := E_1.\text{balio} + T.\text{balio}\} \quad (1)$

$\quad | E - T \quad \{E.\text{balio} := E_1.\text{balio} - T.\text{balio}\} \quad (2)$

$\quad | T \quad \{E.\text{balio} := T.\text{balio}\} \quad (3)$

$T \rightarrow T * F \quad \{T.\text{balio} := T_1.\text{balio} * F.\text{balio}\} \quad (4)$

$\quad | T / F \quad \{T.\text{balio} := T_1.\text{balio} / F.\text{balio}\} \quad (5)$

$\quad | F \quad \{T.\text{balio} := F.\text{balio}\} \quad (6)$

$F \rightarrow \mathbf{osoko} \quad \{F.\text{balio} := \mathbf{osoko.balio}\} \quad (7)$

Atributuak

- Gramatikaren ikur baten atributu bat sintetizatua da analisi-zuhaitzeko adabegi batean duen balioa semeen atributuen balioen arabera kalkulatzeko bada
- Atributu lexikoak
- Bestelako atributuak: atributu heredatuak

3.4 Analisi sintaktikoa

- Goitik beherako analisia
- Analisi aurrusalea
- Analizatzaile aurrusaleen eraikuntza
- Gramatika aurrusalea lortzeko eraldaketa

Goitik beherako analisia

- Analisi-zuhaitza errotik eraikitzen hasi eta hostoetan bukatzen da
- **A** ez-bukaerako ikurrarekin etiketatuko **n** adabegi bakoitzeko
 - » aukeratu **A**-ren produkzio bat eta produkzioaren eskuin aldeko ikurrak izango diren bere **n**-ren semeak eraiki

Adibidea

mota \rightarrow mota_sinplea

| ^ id

| array [mota_sinplea] of mota

mota_sinplea \rightarrow integer

| char

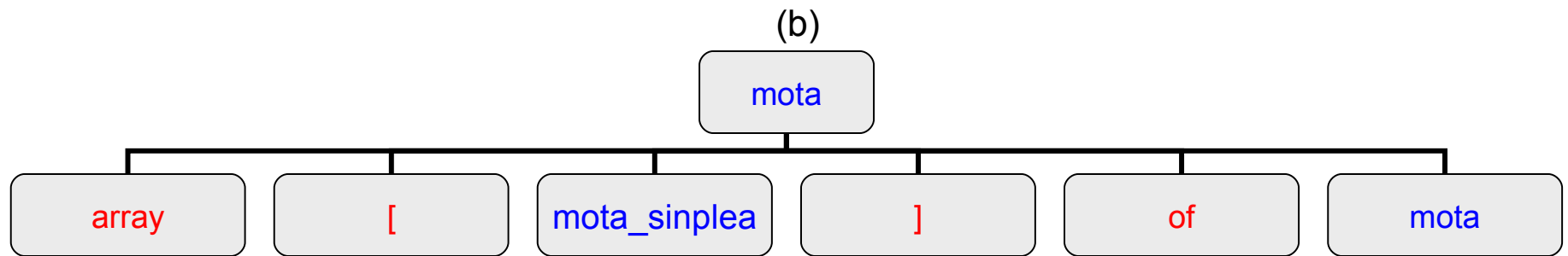
| zenb puntu puntu zenb

array [1..10] of ^ pertsona

(a)

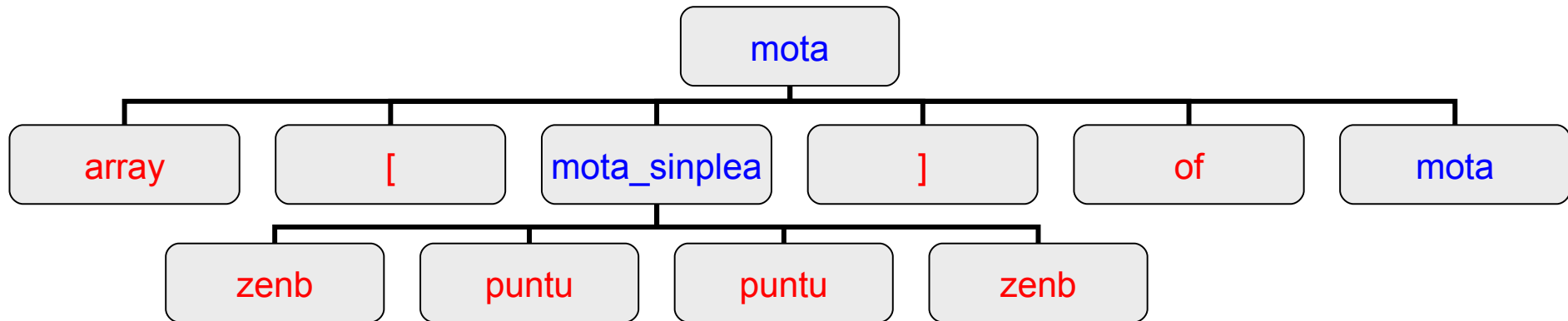
mota

array [1..10] of ^ pertsona

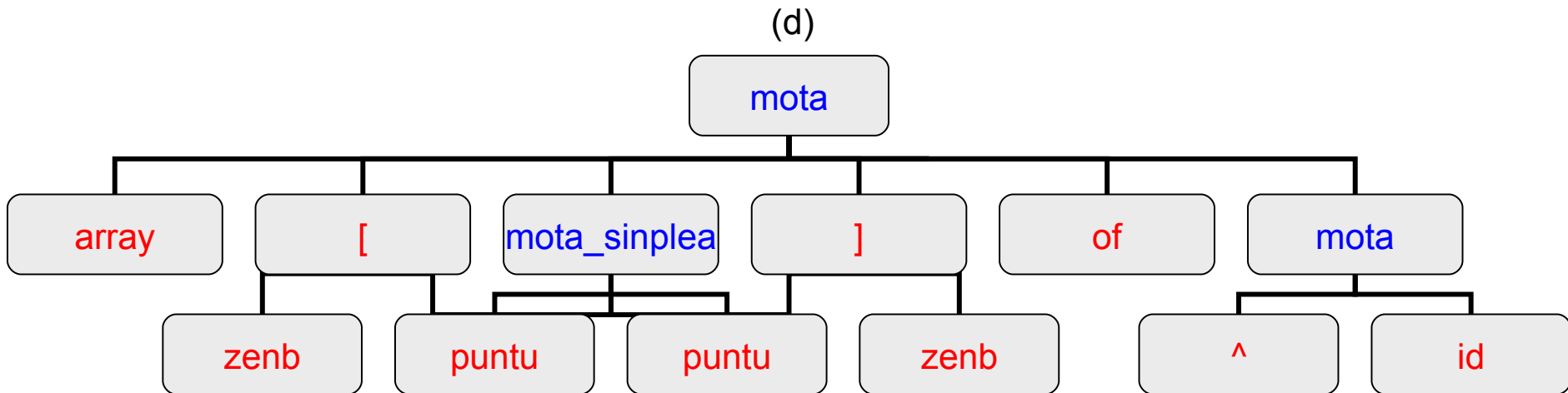


array [1..10] of ^ pertsona

(c)



array [1..10] of ^ pertsona



Analisi aurrealea (I)

- Gramatika batzuentzat, sarrerako token-katea ezkerretik eskuinera korritu ahala joan daiteke zuhaitza eraikitzen aurreko pausoak jarraituaz
- Unean aztertzen gauden tokena *lookahead* deritzo

Analisi aurrealea (II)

- Analisi aurrealean, lookahead horrek determinatzen du anbiguotasunik gabe zein den ez-bukaerako bakoitzerako analisi-prozesuko unean aukeratu behar dugun produkzioa
- **LEHENA**(α): α -k sortzen dituen kateetan ezkerrean azaltzen diren tokenen multzoa (ξ berezi)

Analisi aurrealea (III)

- $A \rightarrow \alpha$
- $A \rightarrow \beta$
- Baldin LEHENA(α) eta LEHENA(β) disjuntuak badira
 - » lookahead-en arabera erabili beharreko produkzioa aukeratu dezakegu

Adibidea

mota \rightarrow mota_sinplea

| ^ id

| array [mota_sinplea] of mota

mota_sinplea \rightarrow integer

| char

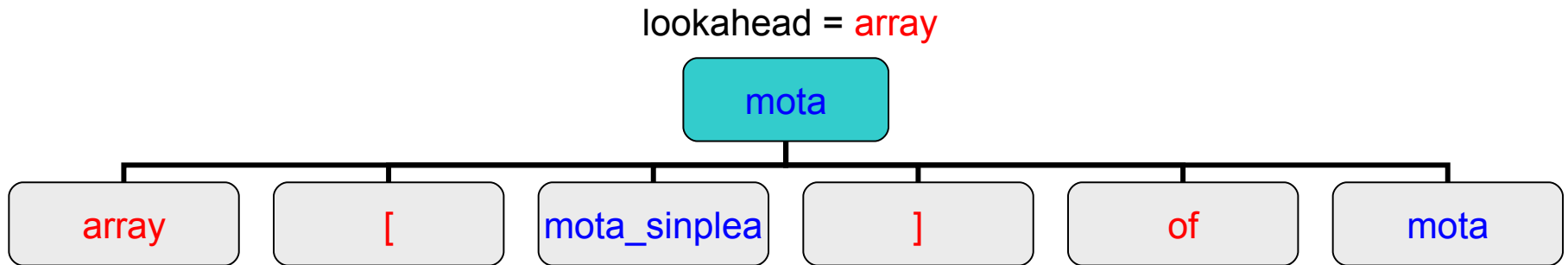
| zenb puntu puntu zenb

array [1..10] of ^ pertsona

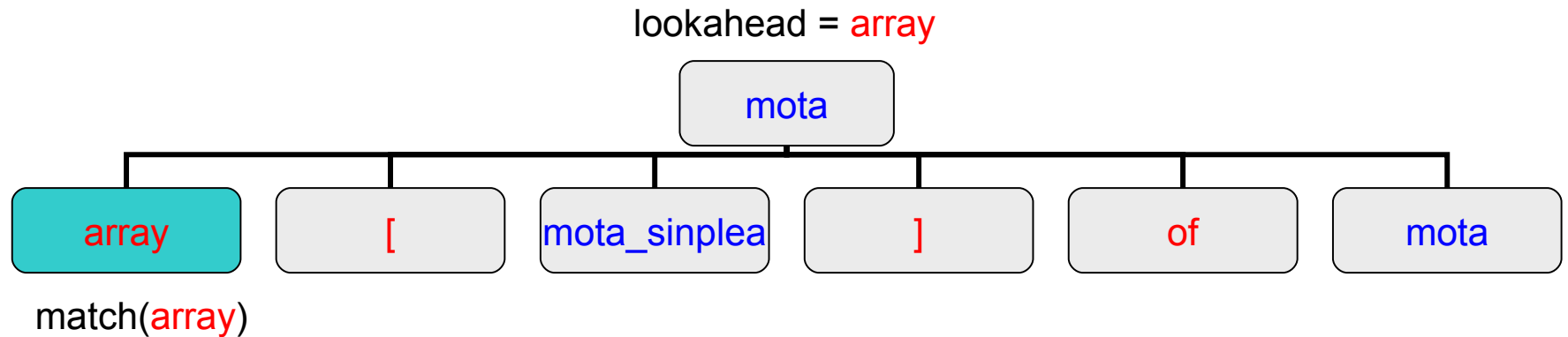
lookahead = array

mota

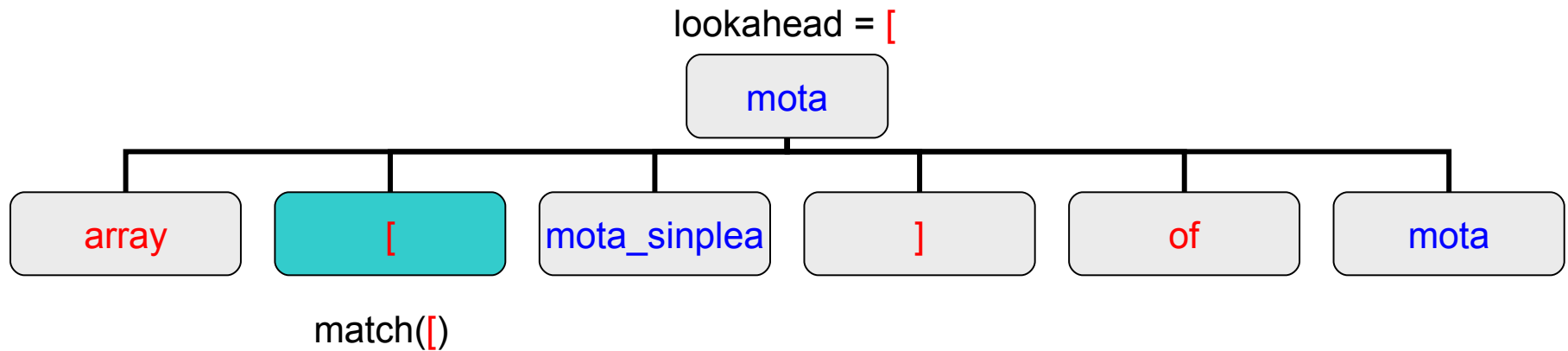
array [1..10] of ^ pertsona



array [1..10] of ^ pertsona

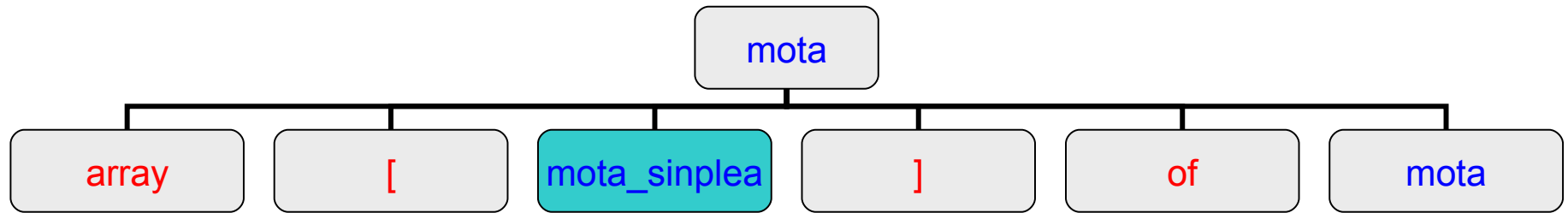


array [1..10] of ^ pertsona

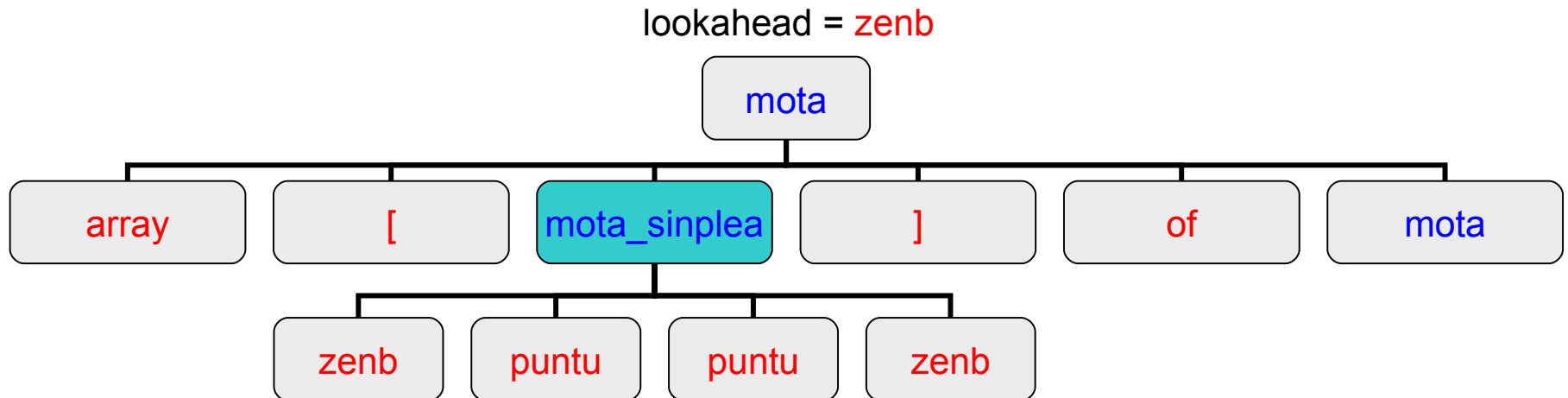


array [1..10] of ^ pertsona

lookahead = **zenb**

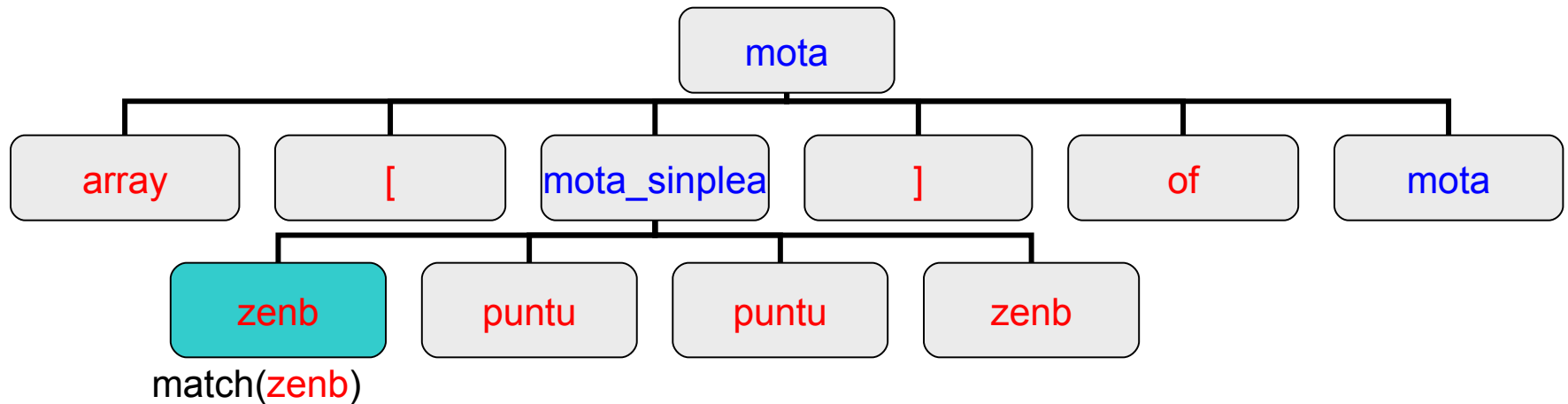


array [1..10] of ^ pertsona

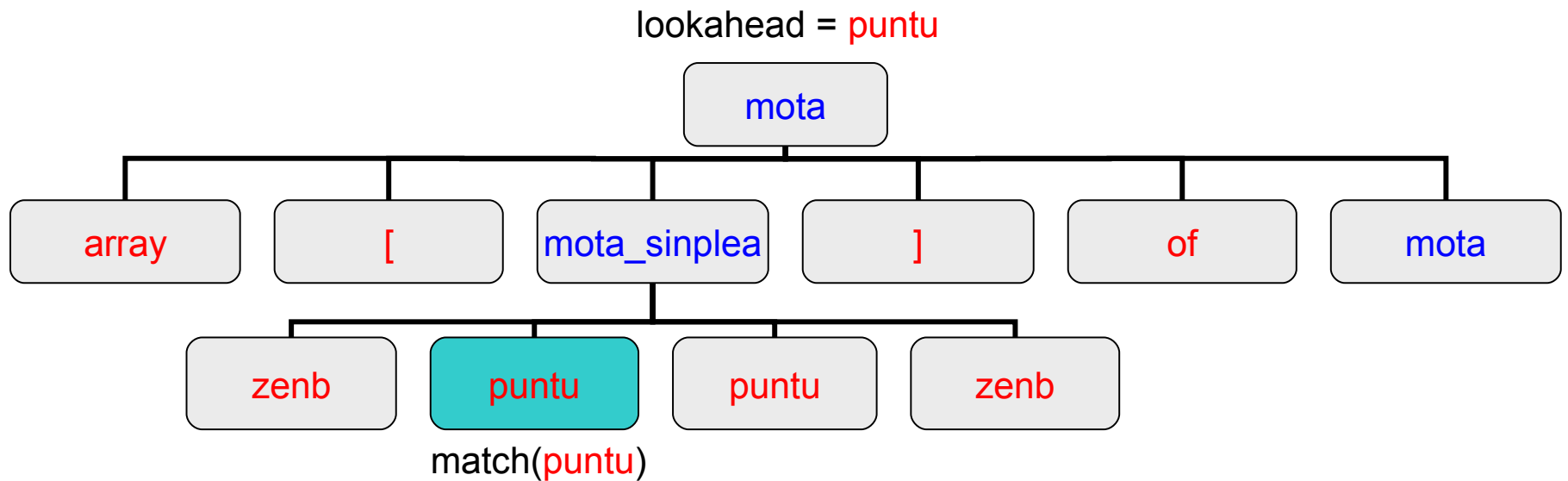


array [1..10] of ^ pertsona

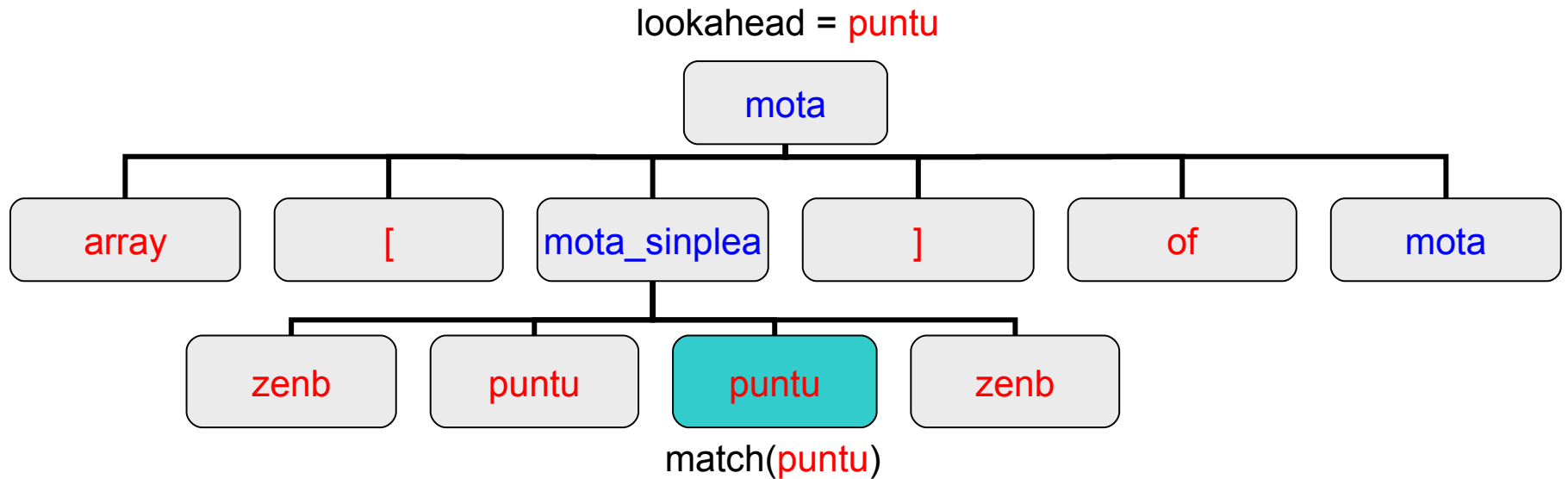
lookahead = **zenb**



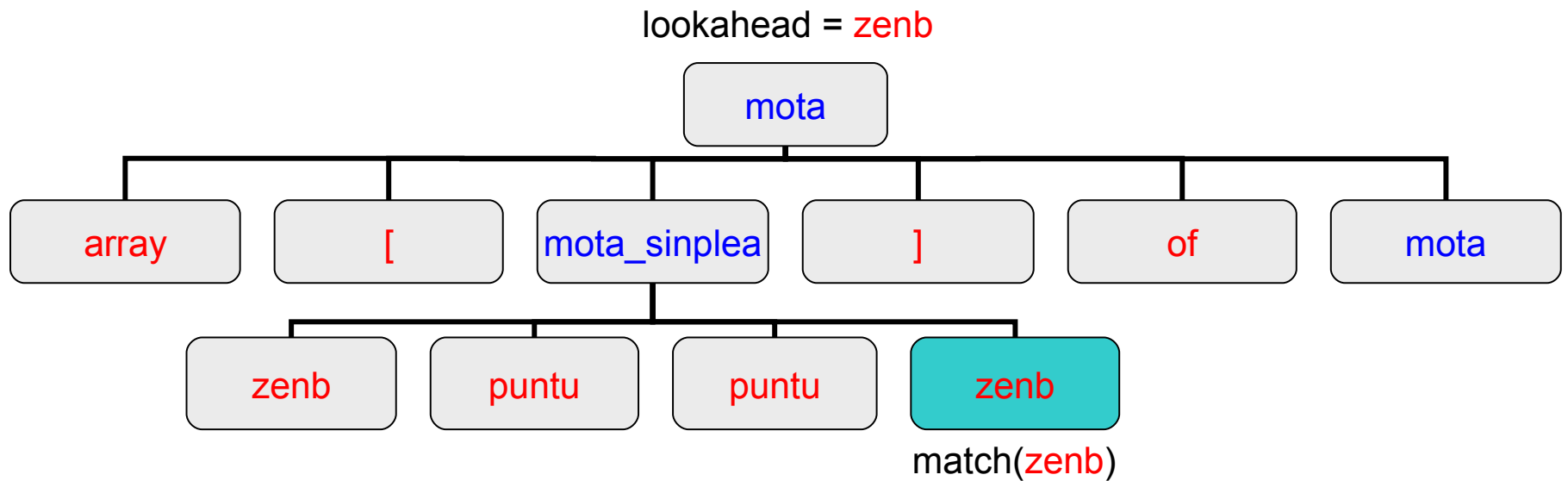
array [1..10] of ^ pertsona



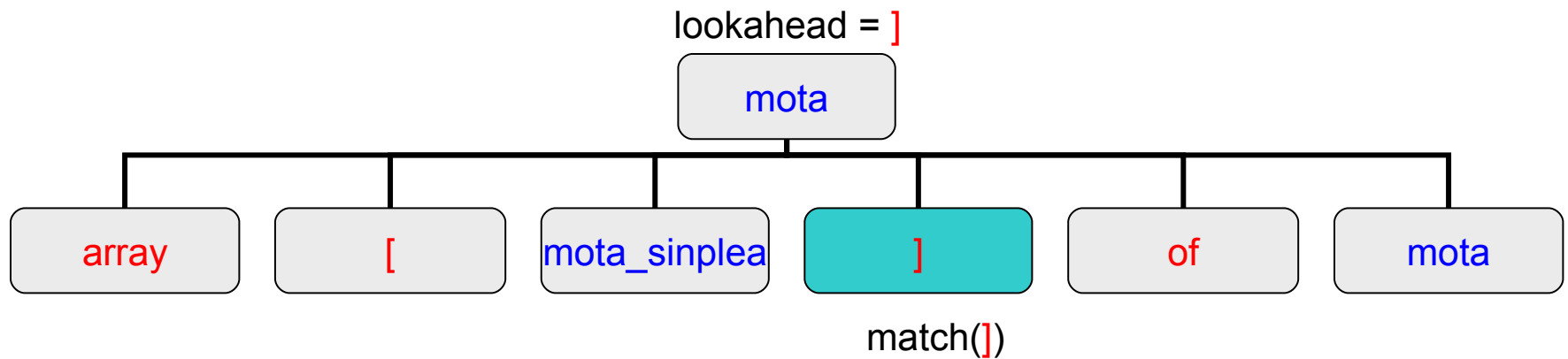
array [1..10] of ^ pertsona



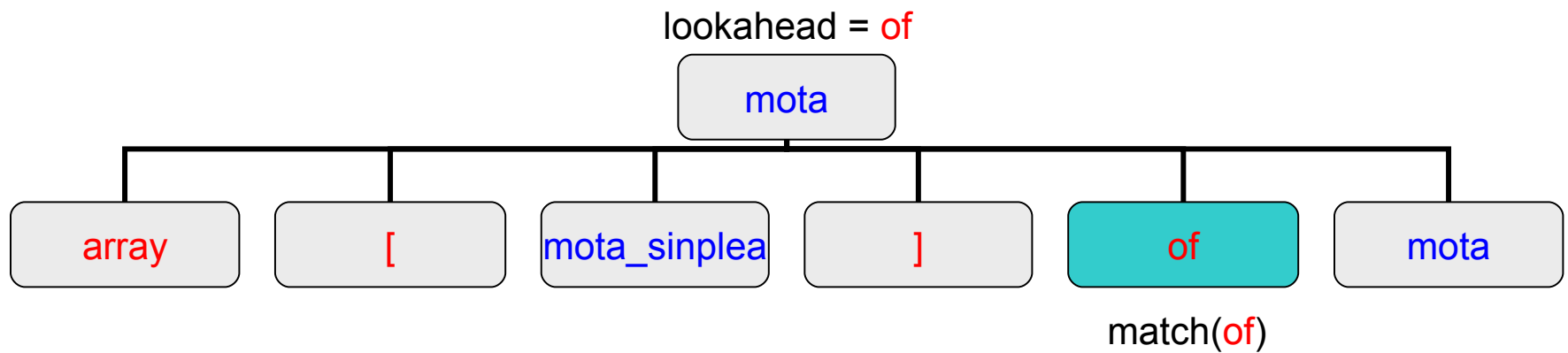
array [1..10] of ^ pertsona



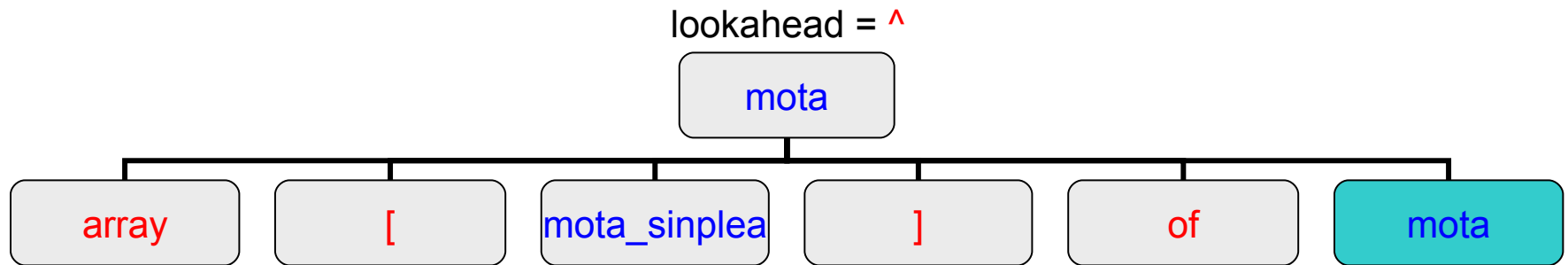
array [1..10] of ^ pertsona



array [1..10] of ^ pertsona

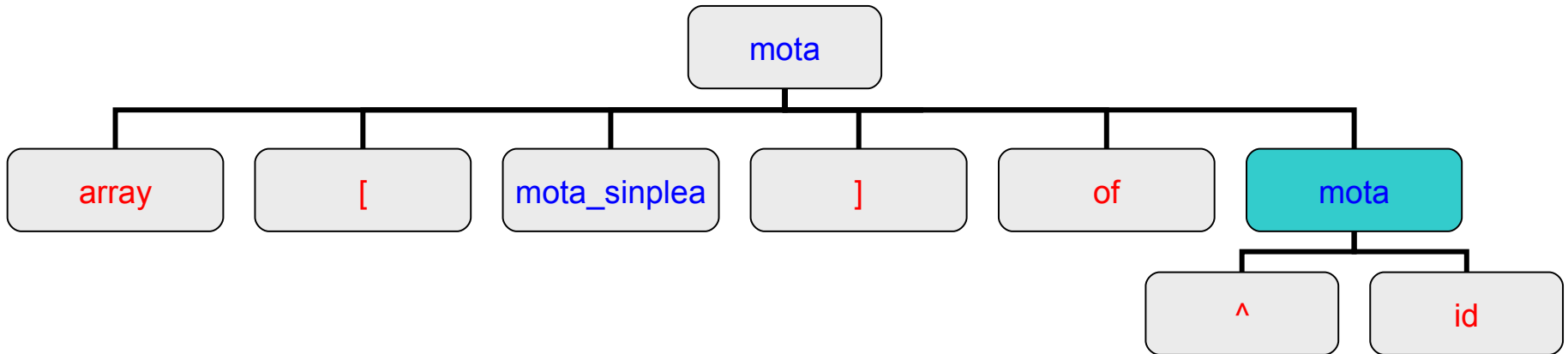


array [1..10] of ^ pertsona

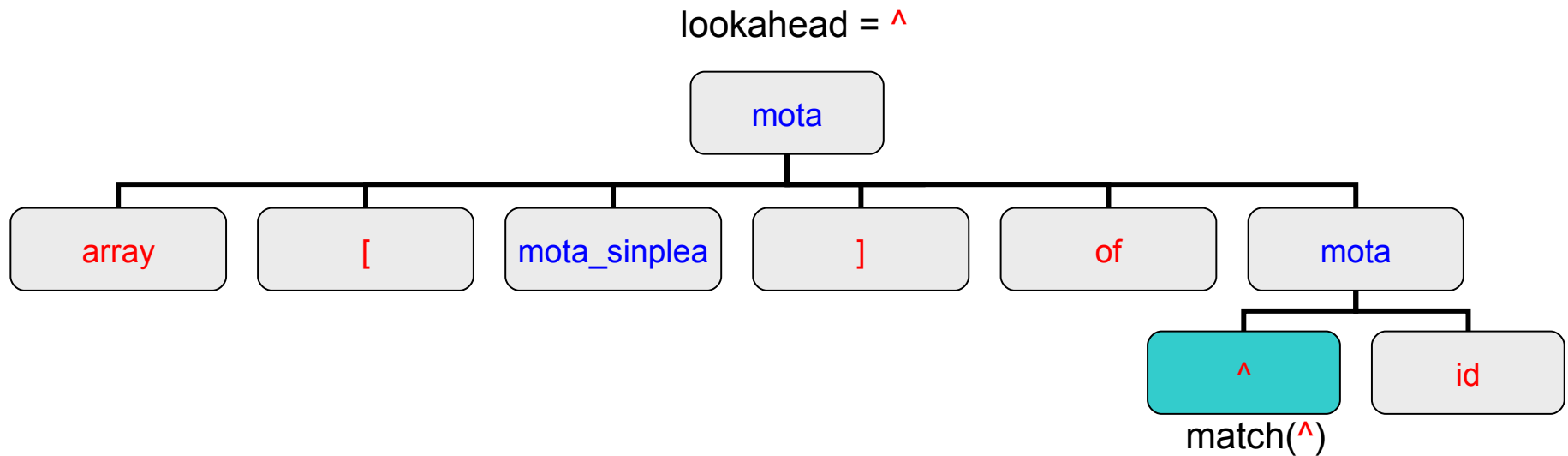


array [1..10] of ^ pertsona

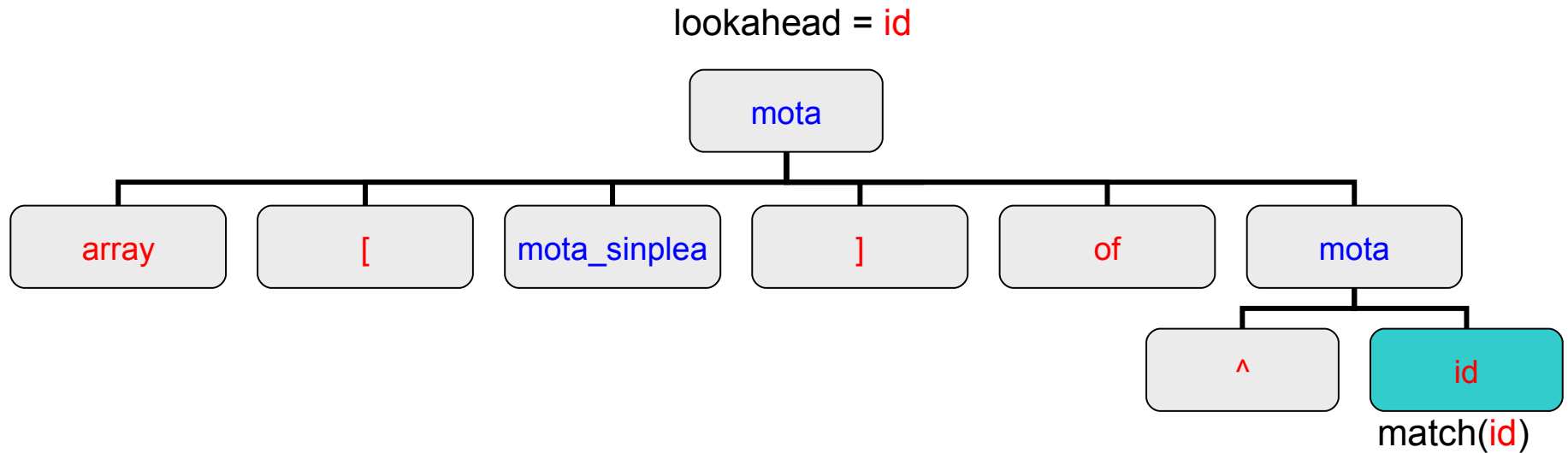
lookahead = ^



array [1..10] of ^ pertsona



array [1..10] of ^ pertsona



Analizatzaile aurrerale baten eraikuntza (I)

- Ez-bukaerako ikur bakoitzeko prozedura bat
 - » Erabaki zein produkzio erabili
 - » Egikaritu produkzioaren eskuin-partearen arabera eraikitako ekintza segida
 - Ez-bukaerakoak: dagokion prozedurari deitu
 - Bukaerakoak: ziurtatu sarrerako tokenarekin parekatzen den (match funtzioa)

Analizatzaille aurrresale baten eraikuntza (II)

```
procedure A;  
begin  
  if lookahead barne LEHENA( $\alpha$ ) then  
    -- analizatu  $\alpha$   
  else if lookahead barne LEHENA( $\beta$ ) then  
    -- analizatu  $\beta$   
  else  
    -- errorea  
  end if;  
end A;
```


Adibidearen inplementazioa

```
procedure match(tokena: token_mota);  
begin  
    if lookahead = tokena  
    then      lookahead := hurrengo_tokena()  
    else      errorea  
end;
```


Adibidearen inplementazioa (II)

```
procedure mota;  
begin  
    if lookahead barne { integer char zenb }  
    then          mota_sinplea  
    else if lookahead barne { ^ }  
    then          match(^); match(id);  
    else if lookahead barne { array }  
    then match(array); match('[');  
        mota_sinplea; match(']');  
        match(of); mota;  
    else errorea  
end;
```


Adibidearen implementazioa (III)

```
procedure mota_sinplea;  
begin  
    if lookahead barne { integer }  
        then    match(integer)  
    else if lookahead barne { char }  
        then    match(char);  
    else if lookahead barne { zenb }  
        then    match(zenb);  
                match(puntu); match(puntu);  
                match(zenb);  
    else errorea  
end;
```


Gramatika aurresalea lortzeko eraldaketa

- Analisi aurresalea onartzen duen gramatika lortzeko pausoak:
 1. Anbiguotasuna kendu
 2. Ezker-errekurtsibitatea kendu
 3. Faktore komunak kendu

Ezker-errekurtsibitatea (I)

- Ezker-errekurtsibitate (zuzena)
gramatikan $A \rightarrow A\alpha$ itxurako erregela dagoenean gertatzen da
- Ezker-errekurtsibitate zuzena balego, ezinezkoa da gramatika horrentzat goitik beherako analizatzaile aurrealea egitea

Ezker-errekurtsibitatea (II)

- $A \rightarrow A\alpha \mid \beta$
non $L(A) = L(\beta) L(\alpha)^n \quad n \geq 0$
- $A \rightarrow \beta A'$
 $A' \rightarrow \xi \mid \alpha A'$
non $A, A' \in N \wedge \alpha, \beta \in (T \cup N)^*$

Faktorizazioa

- Ez-bukaerako baten bi produkziok hasiera bera badute, ezin da goitik beherako analisi aurrealea egin

- Faktorizazioa:

$$A \rightarrow \alpha B$$

$$A \rightarrow \alpha C$$

$$A \rightarrow D$$

non lehena(B) eta lehena(C) disjuntuak

$$A \rightarrow \alpha A'$$

$$A \rightarrow D$$

$$A' \rightarrow B$$

$$A' \rightarrow C$$

Adierazpenak berriz

$$E \rightarrow T E'$$

$$T \rightarrow F T'$$

$$E' \rightarrow + T E'$$

$$T' \rightarrow * F T'$$

$$| - T E'$$

$$| / F T'$$

$$| \xi$$

$$| \xi$$

$$F \rightarrow \mathbf{osoko}$$

$$| \mathbf{id}$$

Analizatzaille aurrresale baten adibidea (I)

```
procedure E is
begin
  if -- lookahead barne { osoko, id } -- then
    T;
    E_LEHENA;
  else
    -- errorea tratatu
  end if;
end E;
```


Analizatzaile aurrresale baten adibidea (II)

```
procedure E_LEHENA is
begin
    if -- lookahead barne {+} -- then
        -- ezkondu +
        T; E_LEHENA ;
    elsif -- lookahead barne {-} -- then
        -- ezkondu -
        T; E_LEHENA;
    else -- ez egin ezer !!
    end if;
end E_LEHENA ;
```


3.5 PLen eraikitzaile nagusien itzulpena

- Adierazpen aritmetikoak
- Adierazpen boolearrak
- Baldintzazko sententziak
- Agindu errepikakorrak

Adierazpen aritmetikoen itzulpena

- Adierazpen aritmetikoen itzultzailea, non osokoak beti bezala edo zenbaki erromatarrez adieraz daitezkeen
- Adibidez: **V - III - 2** adierazpenaren itzulpena honakoa litzateke

t1:= 5 - 3;

t2:= t1 - 2;

SZIEen transformazioaren adibidea (kalkulagailua)

(1) $E \rightarrow E - \mathbf{osoko} \{E.\mathbf{balio} := E_1.\mathbf{balio} - \mathbf{osoko}.\mathbf{balio}\}$
 $\quad | \mathbf{osoko} \quad \{E.\mathbf{balio} := \mathbf{osoko}.\mathbf{balio}\}$

(1) $E \rightarrow \mathbf{osoko} \quad \{E'.\mathbf{hbalio} := \mathbf{osoko}.\mathbf{balio}\}$
 $\quad E' \quad \{E.\mathbf{balio} := E'.\mathbf{balio}\}$

(2) $E' \rightarrow - \mathbf{osoko}$
 $\quad \{E'_1.\mathbf{hbalio} := E'.\mathbf{hbalio} - \mathbf{osoko}.\mathbf{balio}\}$
 $\quad E' \quad \{E'.\mathbf{balio} := E'_1.\mathbf{balio}\}$
 $\quad | \xi \{E'.\mathbf{balio} := E'.\mathbf{hbalio}\}$

Atributuak eta SZIE

- Atributuak lokalak dira:
 - » Produkzioarekiko lokalak = azpizuhaitzarekiko lokalak
- Sintetizatuak:
 - » Balioa **ezkerretara** dituen ez-bukaerakoan atributuetan oinarrituta esleitzen diete
 - » Balioa gurasoaren atributuak jasotzen du
- Heredatuak:
 - » Gurasoarengandik dator
 - » Balioa **eskuinetara** dituen ez-bukaerakoan atributuetara pasatzen du

SZIE

(adierazpenen itzulpena)

(1) $I \rightarrow \mathbf{id} := E ;$

$\{\mathbf{ag_gehitu}(\mathbf{id.izena}||:=||E.izena)\}$

(2) $E \rightarrow E - E$

$\{E.izena:=\mathbf{id_berria}();$

$\mathbf{ag_gehitu}(E.izena||:=||E_1.izena|| - ||E_2.izena)\}$

$| \mathbf{osoko} \{E.izena:= \mathbf{osoko.izena}\}$

$| \mathbf{id} \quad \{E.izena:= \mathbf{id.izena}\}$

?



E-retzako SZIE

(adier. itzul. – ez anbigua)

(2A) $E \rightarrow E - F$

{E.izena:=id_berria();

ag_gehitu(E.izena||:=||E₁.izena|| - ||F.izena||)

| F {E.izena:= F.izena}

(2B) $F \rightarrow \mathbf{osoko} \{F.izena:= \mathbf{osoko.izena}\}$

| id {F.izena:= id.izena}

?

E-rentzako SZIE

(adier. itzul. – beheranzko aurrresalea)

(2A') $E \rightarrow F$ $\{E'.hi := F.izena\}$
 E' $\{E.izena := E'.izena\}$

(2A'') $E' \rightarrow - F$
 $\{E'_1.hi := id_berria();$
 $ag_gehitu(E'_1.hi || := ||E'.hi|| - ||F.izena||)\}$
 E' $\{E'.izena := E'_1.izena\}$
 $| \xi \{E'.izena := E'.hi\}$

SZIE osoa(adier. itzul. - beheranzko aurrealea)

$I \rightarrow \mathbf{id} := E \quad \{\text{ag_gehitu}(\mathbf{id.izena}||:=||E.izena)\} \quad (0)$

$E \rightarrow F \quad \{E'.hi := F.izena\} \quad (1)$

$E' \quad \{E.izena:= E'.izena\} \quad (2)$

$E' \rightarrow -F \quad \{E'_1.hi:=id_berria();$
 $\quad \text{ag_gehitu}(E'_1.hi||:=||E'.hi|| - ||F.izena)\} \quad (3)$

$E' \quad \{E'.izena:= E'_1.izena\} \quad (4)$

$| \xi \{E'.izena:= E'.hi\} \quad (5)$

$F \rightarrow \mathbf{osoko} \quad \{F.izena:= \mathbf{osoko.izena}\} \quad (6)$

$| \mathbf{id} \quad \{F.izena:= \mathbf{id.izena}\} \quad (7)$

E-rentzako SZIE

- Abstrakzio funtzionalak:

- » ag_gehitu
- » id_berria

- Atributuak

- » Heredatuak
 - E'.hi (hizena)
- » Sintetizatuak
 - E.izena
 - E'.izena
 - F.izena

Itzultzaile aurre-sale baten adibidea

- Produkzio baten inplementazioa egiteko:
 - Ezkerretara dagoen ez-bukaerakoari dagokion prozedura definitu
 - Atributu bakoitzeko parametro bat prozeduran:
 - Atributu heredatuak: **in** motakoak
 - Atributu sintetizatuak: **out** motakoak
 - Prozedura barruan produkzioaren eskuin partea inplementatzeko:
 - Ez-bukaerakoan atributu bakoitzeko aldagai lokal bat
 - Ez-bukaerakoak: prozedurari deia (parametro egokiekin)
 - Bukaerakoak: match funtzioari deia (parametro gisa token mota egokiarekin)
 - Erregela semantikoak dagokien posizioan inplementatu

Itzultzaile aurresale baten adibidea (II)

```
procedure E ( E_izena :   out string ) is  
    F_izena, E_Lehena_izena, E_Lehena_hizena : string;  
begin  
    F ( F_izena);  
    E_Lehena_hizena := F_izena;  
    E_Lehena (E_Lehena_hizena, E_Lehena_izena);  
    E_izena := E_Lehena_izena;  
end E;
```


Itzultzaile aurresale baten adibidea (III)

```
procedure E_Lehena (E_Lehena_hizena : in string;  
                    E_Lehena_izena : out string ) is  
    F_izena, E_Lehena_1_izena, E_Lehena_1_hizena : string;  
begin  
    if Token_Mota ( Lookahead ) = T_Kendu then  
        match ( T_Kendu );  
        F ( F_izena);  
        E_Lehena_1_hizena := id_berria;  
        ag_gehitu(E_Lehena_1_hizena & " := " & E_Lehena_hizena & " - " & F_izena));  
        E_Lehena (E_Lehena_1_hizena, E_Lehena_1_izena) ;  
        E_Lehena_izena := E_Lehena_1_izena;  
    else  
        E_Lehena_izena := E_Lehena_hizena;  
    end if ;  
end E_Lehena;
```


Ariketa (I)

- Itxura honetako erazagupenaren itzulpena espezifikatu:

var *id_zerrenda* : **osoko** edo **erreal**

non itzulpena erazagupen sinpleagoen
zerrenda bat den:

id : **osoko** (*edo erreal*)

Ariketa (II)

- Itxura honetako erazagupenaren itzulpena espezifikatu:

osoko: *id_zerrenda EDO*

erreal: *id_zerrenda*

non itzulpena erazagupen sinpleagoen zerrenda bat den:

id : **osoko** (*edo erreal*)

Ariketa (III)

- Espezifikatu esleipen aginduen (*id* := *adierazpena*) segida batek eratzen duen programaren itzulpena, non objektu lengoaia esleipen sinpleez osatua dagoen (bi edo hiru eragile).

Ariketa (IV)

- Espezifikatu honako sententzien itzulpena:
 - » loop
 - » adierazpena
 - » if
 - » while
 - » exit if

Ebazpenak (I.1)

Erazagupenak \rightarrow **var** id_zerrenda : mota ;

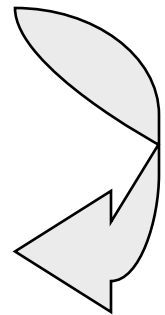
mota \rightarrow **osoko** | **erreal**

id_zerrenda \rightarrow id_zerrenda , **id**
| **id**

id_zerrenda \rightarrow **id** zerrenda_bis

zerrenda_bis \rightarrow , **id** zerrenda_bis

| ξ



Ebazpenak (I.2)

Erazagupenak \rightarrow **var** id_zerrenda : mota;
{ erazagupenak_gehitu(id_zerrenda.izenak,
mota.mota)}

mota \rightarrow **osoko** {mota.mota := 'osoko'}
| **erreal** {mota.mota := 'erreal'}

Ebazpenak (I.3)

$\text{id_zerrenda} \rightarrow \mathbf{id} \text{ zer_bis}$

$\{\text{id_zerrenda.izenak} := \text{gehitu}(\text{zer_bis.izenak}, \mathbf{id.izena})\}$

$\text{zer_bis} \rightarrow , \mathbf{id} \text{ zer_bis}$

$\{\text{zer_bis.izenak} := \text{gehitu}(\text{zer_bis}_1.\text{izenak}, \mathbf{id.izena})\}$

$| \xi \{\text{zer_bis.izenak} := \text{lista_hutsa}()\}$

Ebazpenak (II.1)

Erazagupenak \rightarrow

mota {id_zer.hmota := mota.mota} : id_zer ;

id_zer \rightarrow **id**

{ag_gehitu(id.izena|| : ||id_zer.hmota);
zer_bis.hmota := id_zer.hmota;}

zer_bis

Ebazpenak (II.2)

zer_bis \rightarrow , **id**

{ag_gehitu(**id**.izena|| : ||zer_bis.hmota);
zer_bis₁.hmota := zer_bis.hmota}

zer_bis

| ξ

SZIE (abstrakzioak)

- *erazagupenak_gehitu(zerrenda, mota)*
zerrendako izen bakoitzeko lehenengotik
azkenera mota honetako agindu bat gehituko
du:

izena : mota

- *gehitu(izena,zerrenda) → zerrenda*
sarrerako zerrendan izena **hasieran**
txertatzen du eta zerrenda berria itzultzen du

Adierazpen boolearren itzulpena

$E \rightarrow E \text{ er_erl } E$

```
{ E.true := hasi_lista(lortu_erref());  
  E.false := hasi_lista(lortu_erref() + 1);  
  ag_gehitu(if ||E1.izena|| er_erl.mota ||E2.izena ||goto );  
  ag_gehitu(goto ); }
```


Baldintzazko sententziak (I)

(1) $S \rightarrow \text{if } E \text{ then } M \text{ } S \text{ } N \text{ else } M \text{ } S \text{ } M$

```
{ ag_osatu(E.true, M1.erref);  
  ag_osatu(E.false, M2.erref);  
  ag_osatu(N.next, M3.erref); }
```

(2) $N \rightarrow \xi$

```
{ N.next:=hasi_lista(lortu_erref());  
  ag_gehitu(goto ); }
```


Baldintzazko sententziak (II)

(3) $S \rightarrow \text{if } E \text{ then } M \text{ } S \text{ } M$

```
{ ag_osatu(E.true, M1.erref);  
  ag_osatu(E.false, M2.erref); }
```


Agindu errepikakorrak

(4) $S \rightarrow \mathbf{while} \ M \ E \ \mathbf{do} \ M \ S \ M$
 { ag_osatu(E.true, M_2 .erref);
 ag_osatu(E.false, M_3 .erref + 1);
 ag_gehitu(goto M_1 .erref);}

3.6 Analisi semantikoa

- Egiaztapen estatiko eta dinamikoa
- Moten egiaztatzaile baten espezifikazioa
- Moten baliokidetasun eta bihurketa
- Funtzio eta eragile "gainkargatuak"

Murritzapen semantikoak

- Moten egiaztapenak
- *Kontrol-fluxuaren* gaineko egiaztapena
- Erazagupenaren eta erazagupen bakarraren egiaztapena
- Izenekin lotutako egiaztapenak

Egiaztapen estatiko eta dinamikoa (I)

- Egiaztapen estatikoa **konpilazio**-denboran egiten da
- Egiaztapen dinamikoa **exekuzio**-denboran egiten da (ezin denean estatikoki egin)

Konpiladoreak **egiaztapen dinamikoa egiteko kodea sortu behar du** itzulpenaren zati bat bezala

Egiaztapen estatiko eta dinamikoa (II)

- Programazio-lengoaia baten deskribapenak azaldu behar du zer egin behar den konpilazio-denboran eta zer exekuzio-denboran

Moten egiaztatzailer baten espezifikazioa

- Bi helburu:

- » Lengoaiaren murriztapenen espezifikazioa datu-moten ikuspegitik (moten sistema)
 - Datu-moten arteko erlazioak betetzen dira?
- » Espezifikazio modua: konpiladorean murriztapenak egiaztatuko dituen zatiaren eraikuntza gidatzeko
 - Zer egin errorea gertatzean?

Moten sistemak

- Eragigaien moten gainerako murriztapenak, eragilearen arabera
- Eragiketa baten emaitzaren motaren definizioa
- Murriztapenak estatikoki edo dinamikoki ebaluatu behar diren
 - » Moten sistema "seguruak" edo "strong-typing"
 - Guztia estatikoki egiaztatzen da
 - Ez da egikaritzean motekin lotutako errorerik gertatuko
 - Motak erazagutzera behartzen du

Moten egiaztatzailerik baten adibidea

programa \rightarrow erazagupenak ; adierazpena

erazagupenak \rightarrow erazagupenak ; erazagupenak

erazagupenak \rightarrow **id** : mota

mota \rightarrow **char**

mota \rightarrow **integer**

mota \rightarrow **array** [**osoko**] of mota

mota \rightarrow **access** mota

Eta adierazpenak?

adierazpena → **literal**

adierazpena → **osoko**

adierazpena → **id**

adierazpena → adierazpena **mod** adierazpena

adierazpena → adierazpena [adierazpena]

adierazpena → adierazpena . **all**

Egiaztapena espezifikatzen duen SZIE (I)

programa → erazagupenak ; adierazpena

{baldin adierazpena.mota = 'errore' orduan
errorea_tratatu}

erazagupenak → erazagupenak ; erazagupenak

erazagupenak → **id** : mota

{mota_gehitu(id.izena,mota.mota)}

mota → **char**

{mota.mota := 'karaktere'}

SZIE (II)

mota → **integer**

{mota.mota := 'osoko'}

mota → **array** [**osoko**] of mota

{mota.mota := array(**osoko**.balio,
mota₁.mota)}

mota → **access** mota

{mota.mota := erakusle(mota₁.mota)}

SZIE (III)

adierazpena → **literal**

{adierazpena.mota := 'karaktere'}

adierazpena → **osoko**

{adierazpena.mota := 'osoko'}

adierazpena → **id**

{adierazpena.mota:=mota_lortu(id.izena)}

SZIE (IV)

adierazpena \rightarrow adierazpena **mod** adierazpena

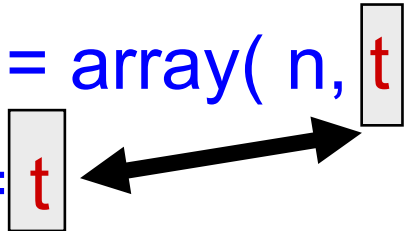
{baldin adierazpena₁.mota = 'osoko' eta

adierazpena₂.mota = 'osoko'

orduan adierazpena.mota := 'osoko'

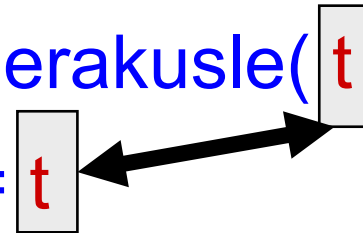
bestela adierazpena.mota := 'errore'}

SZIE (V)

adierazpena \rightarrow adierazpena [adierazpena]
{baldin adierazpena₂.mota = 'osoko' eta
adierazpena₁.mota = array(n, t)
orduan adierazpena.mota := t 
bestela adierazpena.mota := 'errore'}

SZIE (VI)

- `adierazpena → adierazpena . all`
`{baldin adierazpena1.mota = erakusle(t)`
`orduan adierazpena.mota := t`
`bestela adierazpena.mota := 'errore'}`



SZIE (abstrakzioak)

- *mota_gehitu(identifikadorea, mota)*
identifikadorea sinboloen taulan gehituko du
eta sarrerako mota lotuko dio
identifikadorea lehendik erazagutua izan
bada, orduan *errorea* mota esleituko dio
- *mota_lortu (identifikadorea)*
identifikadorea sinboloen taulan badago,
dagokion mota bueltatuko du, eta errorea
mota bestela

Ariketak

- Aurreko espezifikazioa gramatika anbiguo baten oinarrituta eraiki da. Identifikatu anbiguotasuna kentzean zein arazo azaltzen diren eta ebatz itzazu espezifikazioa aldatuz.
- Espezifikazioa aldatu errore-mezu bakarraren ordezt aurkitzen diren errore-semantikoen adina mezu eman daitezten.

Moten baliokidetasun eta bihurketa

- Lengoaia baten moten espezifikazioaren barruan mota **baliokideen** definizioa dago
- Itzulpena egiten den bitartean, askotan moten bihurketarako aginduak sortuko dira

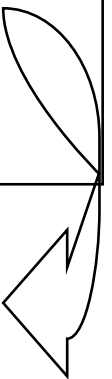
Bihurketa (SZIE)

(1) $A \rightarrow \text{id} := E$

{ ag_gehITU(id.izena|| := ||E.izena) }

(2) $E \rightarrow E + E$

{ E.izena := id_berria();
 ag_gehITU(E.izena|| := ||E₁.izena|| + ||E₂.izena);
}



Bihurketa (SZIE)

(2) $E \rightarrow E + E$

```
{ E.izena := id_berria();  -- batuketaren emaitza kalkulatzeko
  if E1.mota = 'osoko' and E2.mota = 'osoko'
  then
    ag_gehitu(E.izena|| := ||E1.izena||osokoen_batuketa||E2.izena);
    E.mota := 'osoko'
  else if E1.mota = 'erreal' and E2.mota = 'erreal'
  then
    ag_gehitu(E.izena|| := ||E1.izena||errealen_batuketa||E2.izena));
    E.mota := 'erreal'
```


SZIE (II)

else if E_1 .mota = 'osoko' and E_2 .mota = 'erreal'

then

lag1 := id_berria(); -- bihurketa egiteko

ag_gehitu(lag1 || := ||osokotikerrealera|| E_1 .izena);

ag_gehitu(E .izena || := ||lag1||errealen_batuketa|| E_2 .izena));

E .mota := 'erreal'

SZIE (III)

```
else /* E1.mota = 'erreal' and E2.mota = 'osoko' */  
  lag1 := id_berria();  
  ag_gehitu(lag1|| := ||osokotikerrealera||E2.izena);  
  ag_gehitu(E.izena|| := ||E1.izena||errealen_batuketa||lag1));  
  E.mota := 'erreal'  
end if;  
}
```


Funtzio eta eragile gainkargatuak

- gainkarga sintaktiko eta semantikoa
- esleipen dinamikoa eta esleipen estatikoa

3.7 Sinboloen taula

- Sinboloen taularen edukia eta erabilera
- Sinboloen taularen errepresentazioa
- Informazioaren irismena

Sinboloen taularen eduki eta erabilera (I)

- Datu-egitura bat da, konpiladoreak programetan izen batez identifikatuta azaltzen diren objektuei (konstante, mota, aldagai, prozedura, funtzio,...) buruz behar duen informazioa gordetzeko darabilena
- Taulako sarrera bakoitza bikote bat da (izena, informazioa)

Sinboloen taularen eduki eta erabilera (II)

- Sinboloen taula konpiladorearen fase ia guztietan erabiltzen da
- Sinboloen taularen funtzionalitatea definitzeko orduan aukera desberdinak daude. Aukeraketa konpiladorearen diseinatzaileak egin behar du

Sinboloen taularen funtzionalitatea

- Sinboloen taula hasieratu
- Identifikadore bat gehitu
- Begiratu identifikadore bat taulan dagoen
- Gehitu identifikadore batekin lotutako informazioa
- Identifikadore baten informazioa lortu
- Identifikadore bat edo identifikadore-multzo bat taulatik ezabatu

STak eduki ohi duen informazio mota

- Identifikadore bati dagokion karaktere-sekuentzia
- Identifikadore baten atributuak: mota, klasea, ...
- Lotutako parametroak:
 - » Taulak: dimentsio-kopurua eta indizeen balioen mugak
 - » Erregistroak: eremuen izenak eta motak
 - » Prozedurak: parametro-kopurua eta beraien mota ...
 - » Dagokion memoria-posizioaren helbidea

STaren edukia eta erabilera

- Informazioa gehitu eta aztertu egiten da une desberdinetan (askotan)
- Luzera aldakorreko informazioen arazoa kontuan izan behar da (adibidez, array baten izena eta dimentsioak)
- Kasu batzuetan interesgarria izan daiteke sinboloen taularen espazioa berrerrabiltzea
- Prozedura eta funtzioetan:
 - » aktibazio erregistroa

STaren errepresentazioa

- Listak
- Bilaketa zuhaitz bitarrak
- Hash taulak

Listak

- Sinpleena
- Abiadura aldetik motelena
- Inplementazioa: estatikoa edo dinamikoa (arrayak/erakusleak)
- Hobekuntza posibleak: lista ordenatua, azken atzipenaren arabera, ...

Bilaketako zuhaitz bitarrak

- Zuhaitz bitarra orekatua bada eraginkortasuna zerrendaren kasuan baino hobea da, baina konplexuagoa
- Inplementazioa: estatikoa edo dinamikoa
- Zuhaitzak oreka gutxi badu, orduan emaitza listaren soluzioaren antzekoa izango da

Hash taulak

- Diseinu aukera asko
- Eraginkortasun hobe diseinu eta inplementazioaren konplexutasunaren truke
- Inplementazioa: estatikoa edo estatiko/dinamikoa
- Metodoaren eraginkortasuna hash funtzioaren aukeraketaren arabera eta sinonimoen tratamenduaren arabera

Hash taulei buruzko ideia orokorrak (I)

- Egokiak hiztegi motako egiturentzat
- Hash funtzioa:
 - » fhash: gakoa \rightarrow helbidea
- Hashing itxia eta hashing irekia
- Arazo nagusia: hash funtzioak gakoak helbideen artean era uniformearen banatzea datu-egituran

Hash taulak (II)

- n elementutako array bat atzitzeko funtzio errazena (klasikoa)
 - » $H(c) = \text{ord}(c) \bmod n$
- sinonimoak (talka): gako ezberdina edukita hash funtzioaren balio bera dutenak, $H(c1)=H(c2)$

Hash taulak (III)

- talken tratamendu: zerrenda, funtzio osagarriak, sinonimoentzako espazio berezia...
- Bibliografia: Algoritmos + Estructuras de Datos = Programas

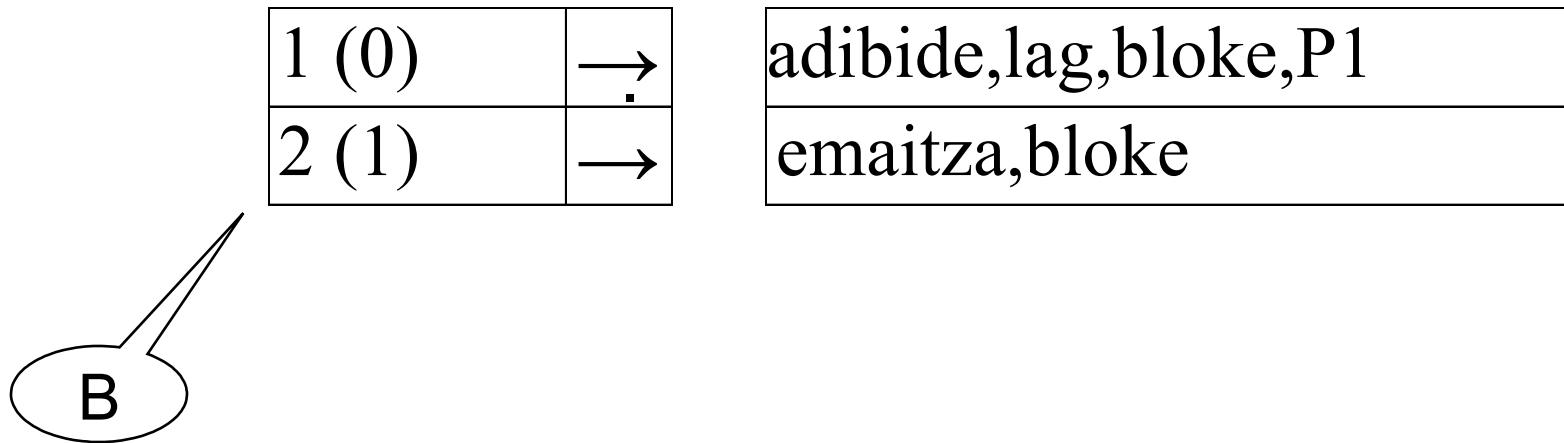
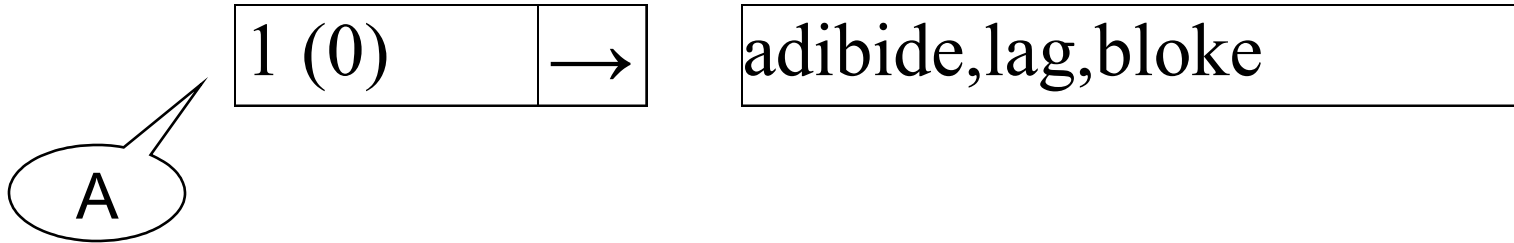
Informazioaren irismena

```
program adibide;
var lag, bloke : integer;
  procedure P1 (var emaitza : integer);
    var bloke : integer;
  begin
    read(bloke);
    emaitza := bloke + lag;
  end;
```

The diagram illustrates the scope of variables in the provided Pascal code. It features three horizontal levels, each marked by a left-pointing arrow and a circled number:

- Level 1:** Indicated by an arrow and a grey oval containing the number '1'. It corresponds to the outermost `program adibide` scope.
- Level 2:** Indicated by an arrow and a grey oval containing the number '2'. It corresponds to the scope of the `procedure P1`. A speech bubble labeled 'A' points to the `var lag, bloke : integer;` line, which is within this scope.
- Level 2':** Indicated by an arrow and an orange oval containing the number '2' with a prime symbol. It corresponds to the scope of the `begin` block inside `procedure P1`. Two speech bubbles labeled 'B' point to lines within this inner scope: one points to `var bloke : integer;` and the other points to `emaitza := bloke + lag;`.

Pilaren egoera

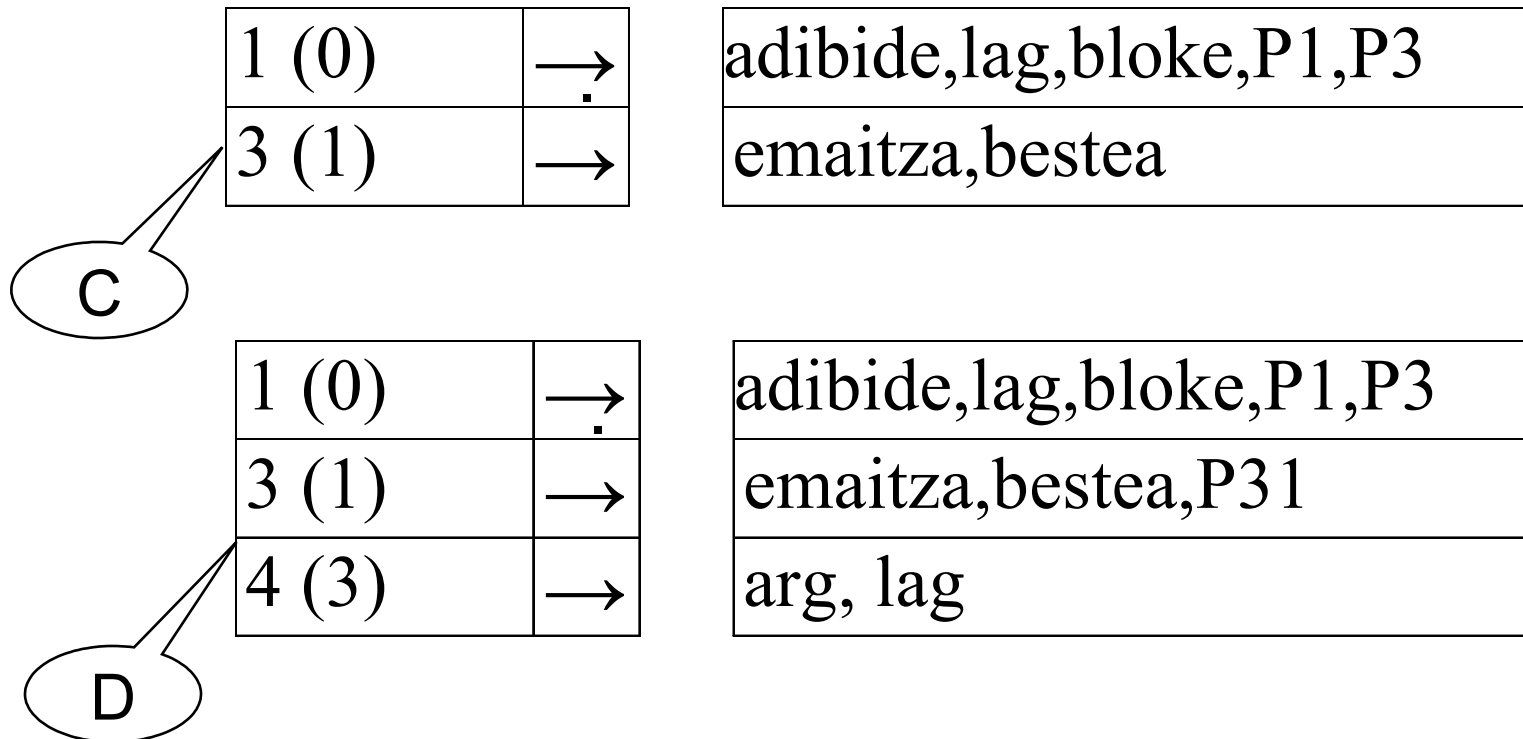


Informazioaren irismena (II)

```

                                ← 3
procedure P3 (var emaitza : integer);
var bestea : integer; C
                                ← 4
    function P31 (var arg : integer): boolean;
    var lag : integer;
    begin D
        read(lag);
        P31 := (lag mod arg) = bloke;
    end;                                ← 4'
```


Pilaren egoera (II)



Informazioaren irismena (III)

begin -- P31en hasiera

E

bestea := bloke;

emaitza := P31(bestea);

end;

begin -- programa nagusia

F

read(lag); bloke:= lag*2;

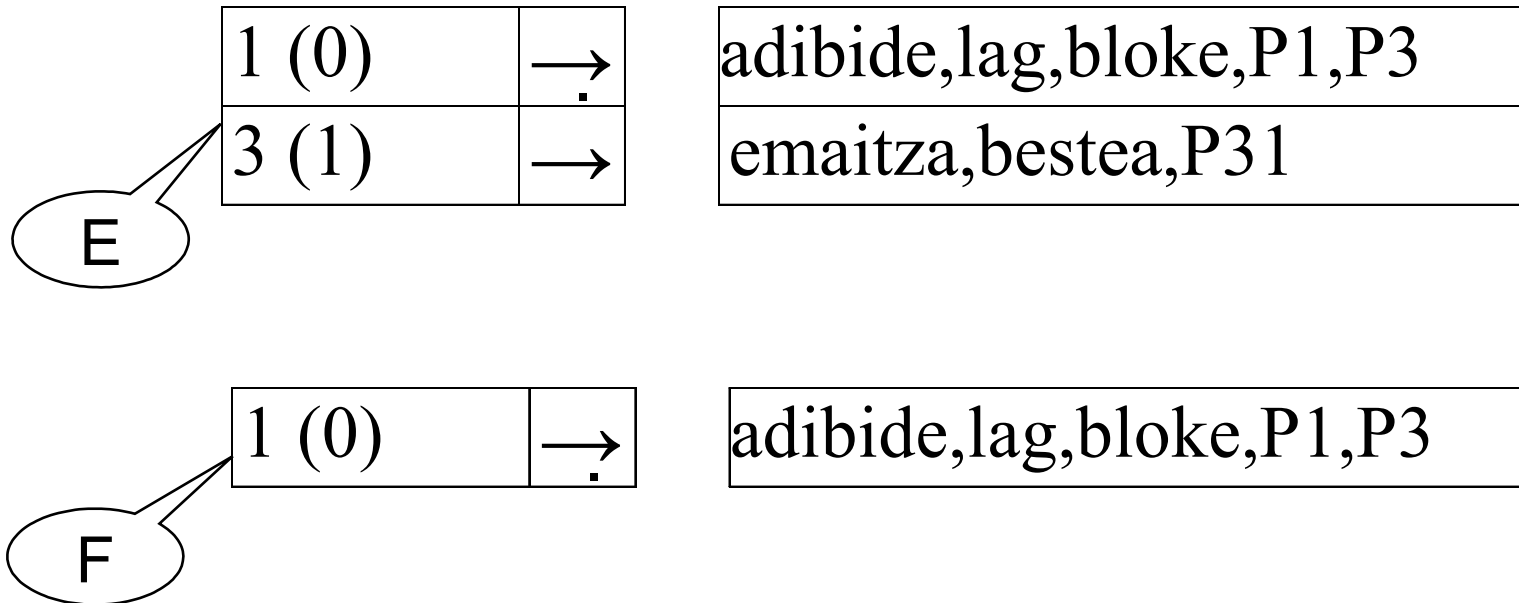
P3(bloke);

end;

1'

3'

Pilaren egoera (III)



3.8 Bitarteko kode motak

- Notazio postfixua, zuhaitz sintaktikoak eta AST-ak
- Hiru helbideetako kodea
- Piladun makina abstraktuak
 - » JVM
 - » p-code

Java Virtual MachineTM

- Garapena Javarekin batera, baina beste lengoaien itzulpenean ere erabil daiteke
- Gehienez 256 agindu (byte 1 koderako). Orain 220 agindu.
- Ez "ortogonal"

Makina birtualaren egitura

- **pc** erregistroa (program counter)
- Pila (bat azpiprozesu bakoitzeko - *thread*)
- Frame-ak (aktibazio-erregistroak)
- Heap
- Koderako zatia

Oinarrizko datu-motak

- **byte** (1 byte)
- **short** (2 byte)
- **int** (4 byte)
- **long** (8 byte)
- **char** (2 byte, Unicode bertsioa)
- **float** (4 byte)
- **double** (8 byte)
- **returnAdress** (ez da Javaren mota bat)
- **reference** (null balio berezia barne)

Agindu-motak (I)

- **Karga eta gordetzeko, eragigaien pila eta aldagai lokalen zatiaren artean**
 - » Tload (karga), Tstore (gordetzeko tokia),...
- **Agindu aritmetikoak**
 - » Batuketa, kenketa, biderkaketa, zatiketa, hondarra, gehikuntza,...
- **"Agindu aritmetikoak "**
 - » desplazamendua, or, and, xor
- **Moten bihurketarako aginduak**
 - » i2l (osokotik osoko luzera - int to long)

Agindu-motak (II)

- **Objektuen sorkuntza eta erabilera**
 - » klaseen eta arrayen instantziak sortu
- **Metodoen deiak eta return-ak**
 - » invoke eta Treturn
- **Jauzietarako aginduak**
 - » goto (baldintzagabea)
 - » if<bald>, if_icmp<bal> (baldintzatua)
- Jauzia aukera desberdinekin: tableswitch, lookupswitch

Motak eta eragiketak (adibidea)

kodea	byte	short	int	long	float	double
Tipush	X	X				
Tconst			X	X	X	X
Tload			X	X	X	X
Tstore			X	X	X	X
Tinc			X			
Taload	X	X	X	X	X	X
Tastore	X	X	X	X	X	X
Tadd			X	X	X	X
...						
i2T	X	X		X	X	X
l2T			X		X	X
...						
Tcomp				X		
...						
if_TcmpOP			X			
...						

Agindu baten deskribapena (I)

- **if_icmp<bald>**

- **Eragiketa**

» Jauzia egin osokoen arteko konparazioa egiazkoa bada.

- **Formatua**

if_icmp<cond>
branchbyte1
branchbyte2

Agindu baten deskribapena (II)

- **Formak**

- » if_icmpeq, if_icmpne, if_icmplt, if_icmpge, if_icmpgt, if_icmple

- **Pila** ..., balio1, balio2 → ...

- **Deskribapena** Bai balio1 zein balio2 *int* motakoak izan behar dira. Biak pilatik aterako dira eta konparatuko dira ...

Adibidea

*The Java Virtual Machine Specification
(Addison-Wesley). orr. 341*

```
void spin() {  
    int i;  
    for (i = 0; i < 100; i++) {  
        ;      // begiztaren gorputza hutsa)  
    }  
}
```


Adibidearen itzulpena

<i>0 iconst_0</i>	// 0 konstante osoko pilan sartu
<i>1 istore_1</i>	// Gailurretik 1 posiziora (i=0)
<i>2 goto 8</i>	// Jauzia for-aren aginduetara
<i>5 iinc 1 1</i>	// 1 h.ko edukia 1ekin gehitu (i++)
<i>8 iload_1</i>	// 1 posizioko edukia (i) pilaratu
<i>9 bipush 100</i>	// 100 konstantea pilaratu
<i>11 if_icmplt 5</i>	// if (i < 100) 5era jauzia egin
<i>14 return</i>	// return void

P-code

- Pascal konpiladoreetan zabaldua
- 130 aginduk osatua
- Motak: helbidea, boolearra, karaktere, osoko, erreal eta multzo
- *Pascal Implementation, S. Pemberton*

Agindu batzuk (*p-code*)

<i>abi</i> (i)	i	Balio absolutua
<i>adr</i> (r,r)	r	Errealen batura
<i>chk</i> (i)	c	Karakterera bihurtu
<i>csp</i> (r,r)	r	Prozedurari deia
<i>equ</i> (x,x)	b	Berdintasuna
<i>not</i> (b)	b	Desberdintasuna
<i>sto</i>	Stop	

3.9 PLen eraikitzaile nagusien itzulpen hedatua

- Erazagupenak
- Esleipen-aginduak
- Adierazpen boolearrak
- Taulen erreferentziak

Erazagupenak (I)

(1) $P \rightarrow \{ \text{desplazamendua} := 0 \} E ; S$

(2) $E \rightarrow E ; E$

(3) $E \rightarrow \text{id_erazag} : \text{Mota}$

$\{ \text{Sartu}(\text{id_erazag.izena}, \text{Mota.mota},$
 $\text{desplazamendua});$
 $\text{desplazamendua} := \text{desplazamendua}$
 $\quad + \text{Mota.tamaina} \}$

Erazagupenak (II)

(4) Mota \rightarrow **integer**

{ Mota.mota := osoko; Mota.tamaina := 4 }

(5) Mota \rightarrow **real**

{ Mota.mota := erreal; Mota.tamaina := 8 }

(6) Mota \rightarrow **array** [**osoko**] of Mota

{ Mota.mota := array(**osoko**.bal, Mota₁.mota);
Mota.tamaina := Mota₁.tamaina * **osoko**.bal }

Erazagupenak (III)

(7) Mota \rightarrow **access** Mota

{ Mota.mota := erakuslea(Mota₁.mota);
Mota.tamaina := 4 }

(8) id_erazag \rightarrow **id**

{ if Erazagutua(id.izena) then
 Errorea();
id_erazag.izena := id.izena }

Esleipen-aginduak

$S \rightarrow id := E$

- *Esleipenak izenekin*

```
{ if Bateriaezinak(Mota_Lortu(id.izena), E.mota)
  then Errorea_Tratatu();
  else ag_gehitu(id.izena|| := ||E.izena) }
```

- *Esleipenak helbideekin*

```
{ if ... then Errorea_Tratatu();
  else
  ag_gehitu(Helbidea_Lortu(id.izena)|| := ||E.hel) }
```


Adierazpen boolearren itzulpena

- B.K.an boolearrak daudenean:
 - » Aritmetikoen antzera, 0/1 balioak kalkulatzen
- B.K.an boolearrik ez dagoenean:
 - » adierazpena osorik ebaluatu behar denean
 - Kodea sortu behar da, adierazpenaren emaitza kalkulatzeko 0/1 balioak erabiliz
 - » adierazpena osorik ebaluatu behar ez denean
 - Ez da koderik sortu behar, adierazpenaren emaitza kodearen posizioaren arabera
 - true eta false atributuak erabili eta ondoren aginduak osatu (backpatching)
 - Praktikarako teknika hau erabili behar da

Adierazpen boolearrak (I)

$E \rightarrow E \text{ or } E$

$E \rightarrow E \text{ and } E$

$E \rightarrow \text{not } E$

$E \rightarrow (E)$

$E \rightarrow \text{id}$ *(mota boolearra dagoenean)*

$E \rightarrow E \text{ er_erl } E$ *(eragile erlazionala)*

Adierazpen boolearrak (II)

Itzulpen numerikoa

- Egiazko/Faltsua balioak errepresentazio numerikora itzuli

$E \rightarrow E \text{ er_erl } E$

```
{ E.izena := id_berria();  
  ag_gehitu(if ||E1.izena ||er_erl.katea ||E2.izena  
            || goto ||lortu_erref() + 3);  
  ag_gehitu(E.izena|| :=|| 0);  
  ag_gehitu(goto ||lortu_erref() + 2);  
  ag_gehitu(E.izena||:= ||1); }
```


Adierazpen boolearrak (III)

Itzulpen numerikoa

$E \rightarrow E \text{ or } E$

```
{ E.izena := id_berria();  
  ag_gehitu(E.izena|| := ||E1.izena ||or ||E2.izena); }
```

ala

$E \rightarrow E \text{ or } E$

```
{ E.izena := id_berria();  
  ag_gehitu(E.izena|| := ||E1.izena|| + ||E2.izena); }
```


Adierazpen boolearrak (IV)

Jauzi bidezko itzulpena

Sententzien erreferentzien listak erabiltzeko:

1. **hasi_lista(i)** *lista berria sortuko du i balioarekin (sententzia baten erreferentzia)*
2. **bildu(L1, L2)** *L1 eta L2 listak emanda, bien bildura bueltatuko du*
3. **ag_osatu(L, Helb)** *Helb helbidearekin L listan adierazitako jauzi aginduak osatuko ditu*

Adierazpen boolearrak (V)

Jauzi bidezko itzulpena

$M \rightarrow \xi$

{ M.erref := lortu_erref(); }

$E \rightarrow E \text{ or } M E$

{ ag_osatu(E_1 .false, M.erref);

E .true := bildu(E_1 .true, E_2 .true);

E .false := E_2 .false; }

Adierazpen boolearrak (VI)

Jauzi bidezko itzulpena

$E \rightarrow E \text{ and } M E$

```
{ ag_osatu(E1.true, M.erref);  
  E.false := bildu(E1. false, E2.false);  
  E.true := E2.true; }
```


Adierazpen boolearrak (VII)

Jauzi bidezko itzulpena

$E \rightarrow \text{not } E$

$\{ E.\text{false} := E_1.\text{true};$
 $E.\text{true} := E_1.\text{false}; \}$

$E \rightarrow (E)$

$\{ E.\text{false} := E_1.\text{false};$
 $E.\text{true} := E_1.\text{true}; \}$

Adierazpen boolearrak (VIII)

Jauzi bidezko itzulpena

$E \rightarrow E \text{ er_erl } E$

```
{ E.true := hasi_lista(lortu_erref());  
  E.false := hasi_lista(lortu_erref() + 1);  
  ag_gehitu(if ||E1.izena ||er_erl.mota ||E2.izena ||goto );  
  ag_gehitu(goto ); }
```


Jauzietarako aginduak

- Etiketak eta goto aginduak kudeatzeko

$S \rightarrow \text{etiketa} : S$

$\text{etiketa} \rightarrow \mathbf{id}$

{ Gorde_Etiketa(id.izena,lortu_erref()); }

$S \rightarrow \mathbf{goto id}$

{ e :=Lortu_Etiketaren_Helbidea(id.izena);
ag_gehitu(goto e); }

Taulen erreferentziak

Hartutako aukera:

- * Taulen elementuen erreferentziak bitarteko kodean erabaki:
 - » honek arrayen elementuen atzipenaren optimizazioa onartzen du, objektu-lengoiarekiko era independentean

Hasierako suposizioak

- Arrayen erreferentzien itzulpena sinplifikatzeko ondoko aspektuak kontsideratuko ditugu:
 - * arrayen mugak konpilazio-denboran eza gutuko ditugu
 - * array baten elementu bakoitzak hitz bat okupatuko du
 - * objektu-makinak memoria hitzetan dauka antolatuta

Adibidea

- * A: array(10, 20) of Integer;
- * Memorian arrayaren elementuek ondoz-ondoko posizioak hartuko dituzte, era honetan:
 $A[1, 1], A[1, 2], \dots, A[10, 19], A[10, 20]$
- * Bedi a A arrayari dagokion memoria-blokearen lehenengo hitzaren posizioa, orduan $A[i, j]$ elementuaren posizioa ondokoa da:
$$a + 20 * (i\text{-ren balioa} - 1) + (j\text{-ren balioa} - 1) =$$
$$(a - 21) + 20 * i\text{-ren balioa} + j\text{-ren balioa}$$

Adibidea - Itzulpena (I)

$A[i, j]$ elementuaren balioa aldagai lagungarri batean lortzeko kodea hau da:

i -ren balioa $t1$ -en kalkulatzeko kodea

$t2 := 20 * t1$

j -ren balioa $t3$ -en kalkulatzeko kodea

$t4 := t2 + t3$

$t := (a - 21) [t4]$

a : A arrayaren espazioa hasten den helbidea

Adibidea - Itzulpena (II)

Aren helbidea exekuzio-denboran
baino ezagutu ezin denean:

i-ren balioa t1-en kalkulatzeko kodea

$t2 := 20 * t1$

j-ren balioa t3-en kalkulatzeko kodea

$t4 := t2 + t3$

$t5 := \text{addr}(A)$

$t6 := t5 - 21$

$t5 := t6[t4]$

Orokortzea n dimentsiotarako

- * $A(d_1, d_2, \dots, d_n)$
- * Osagai-kopurua: $d_1 * d_2 * \dots * d_n$
- * Atzipena $A[i_1, i_2, \dots, i_n]$ elementura
- * $a + (i_1\text{-en balioa} - 1) d_2 * \dots * d_n$
+ $(i_2\text{-ren balioa} - 1) d_3 * \dots * d_n + \dots$
+ $(i_n\text{-ren balioa} - 1)$
- * “Compiladores ...” (Aho, Sethi, Ullman), 496-500
- * “Compilers...” (Aho, Lam, Sethi, Ullman), 381-385

Boolearrak praktikarako

- Praktikan **EZ DAGO MOTA BOOLEARRA**
- AND, OR eta NOT eragileak gehitu behar dira
- Kortozirkuitua aplikatu behar da:
 - Adierazpen osoa ez ebaluatu beharrezko ez denean
 - .TRUE eta .FALSE atributuak erabili

Ariketa

- Praktikako gramatikan eragiketa boolearrak gehitu
- AND, OR eta NOT eragileak gehitu behar dira
- Adierazpen boolear sinpleak ($>$ $<$ $>=$ $<=$...) mantendu behar dira
- Gramatika aurrealea behar da
 - Anbiguotasunik ez
 - eragileen lehentasunak: NOT, AND eta azkenik OR

Boolearrak praktikarako: hasierako egoera

Adi_bool \rightarrow $E > E$

| $E < E$

...

| $E \geq E$

| $E \leq E$

$E \rightarrow E - T$

| $E + T$

| T

$T \rightarrow T * F$

| T / F

| F

$F \rightarrow \text{id}$

| **zbki**

| **(E)**

Ebazpena (I)

Gramatikaren hedapena
(and, or, not):

E_B	→	E_B or T_B
		T_B
T_B	→	T_B and F_B
		F_B
F_B	→	not E_B
		Adi_bool
		(E_B)

Aurretik geneukana berdin:

Adi_bool	→	E > E
		E < E
		...
		E >= E
		E <= E
E	→	E - T
		E + T
		T
T	→	T * F
		T / F
		F
F	→	id
		zbki
		(E)

Ebazpena (II)

Parentesien arazoa / not-en lehentasuna

E_B	→	E_B or T_B	Adi_bool	→	E > E
		T_B	...		
T_B	→	T_B and F_B			E <= E
		F_B			E
F_B	→	not F_B	E	→	E - T
		Adi_bool			...
			F	→	id
					zbki
					(E_B)

Gramatika honek aurrekoak baina
lengoaia zabalagoa definitzen du!!



Adierazpen oker gehiago sortzea ahalbideratzen
du

Ebazpena (III)

Gramatika errazago eta gainera anbiguotasunik gabe

Adi_bool	→	E > E	T	→	T * F
		E < E			T / F
		...			T and F
		E >= E			F
		E <= E	F	→	id
		E			zbki
E	→	E - T			not F
		E + T			(Adi_bool)
		E or T			
		T			

- Murriztapen semantikoak gehitu behar dira erabilera egokia bermatzeko
- Adierazpen aritmetikoen eta boolearren atributu GUZTIAK eraman behar dira ez-bukaerako guztietan