

2. gaia: Analisi lexikoaren fasea

2.1 Sarrera

2.2 Oinarrizko kontzeptuak

2.3 Analizatzaile lexiko baten diseinu
eta inplementazioa

2.4 Analizatzaile lexikoen
espezifikaziorako lengoaia: LEX

2.1 Sarrera

- Sarreratik "balio sintaktikorik" ez duena iragazten du:
 - » Komentarioak
 - » Zuriuneak
- Errore lexikoak detektatu eta tratatzen ditu
- "Esanahia duten" unitate lexikoak (token) identifikatzen ditu.
- Gramatikako bukaerako ikurren (tokenen) atributuen balioa kalkulatzen du.

2.1 Sarrera

- Tokenak eta beraien atributuak
- Konpiladoreko gainontzeko moduluekin duen harremana
- Iturburu-lengoiaren zenbait ezaugarrik sortzen dituzten arazoak
- Errore lexikoak

Oinarrizko tokenak eta bere atributuak (I)

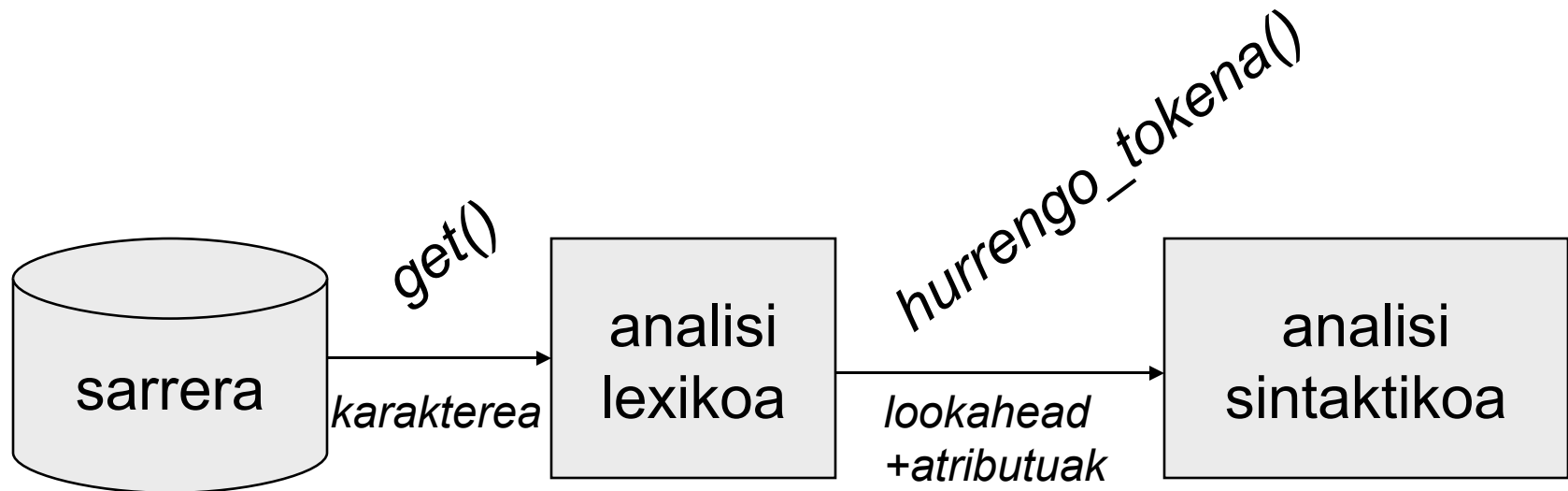
- Identifikadoreak (izena:string)
- Hitz gako/erreserbatuak (izena:string, mota: enum)
- Konstanteak
 - » osokoak (izena: string, balio: osokoa)
 - » errealak (izena: string, balio: erreala)
 - » string-ak, karaktereak

Oinarrizko tokenak eta bere atributuak (II)

- banatzaileak (izena: string, mota: enum)
 - » , :: () []
- eragileak (izena: string, mota: enum)
 - » + - / *
 - » := =
- oharrak

Analisi sintaktikoaren interfazea

- lookahead
- hurrengo_tokena()



Errore lexikoak

- 3.a
- 3.5e-
- “ itxi gabeko string-a
- 3ident := 5
- Karaktere okerra

2.2 Oinarrizko kontzeptuak

- Espezifikazioa
 - Adierazpen erregularrak eta definizio erregularrak
- Eredu ezagutzaileak: Automata finituak
- Adierazpen erregular eta automata finituen arteko baliokidetasun eta bihurketa

2.2 Oinarrizko kontzeptuak

- Σ alfabetoa \rightarrow ikurren multzoa
- Hitza $\rightarrow \Sigma$ alfabetoko ikurren segida
- Σ^* $\rightarrow \Sigma$ gaineko hitz posible guztien multzoa
- Hitzen gaineko eragiketak
- Lengoaia $\rightarrow \Sigma^*$ -en azpimultzoa
- Lengoaiei arteko eragiketak \rightarrow bildura, ebakidura, kateamendua, potentzia, itxidura ...

Lengoaia erregularrak

- Bedi Σ alfabetoa:
 - I. $\{\epsilon\}$ lengoaia erregularra da
 - II. $\{a\}$ lengoaia erregularra da baldin a barne Σ
- Bitez A eta B Σ gaineko bi lengoaia erregular:
 - III. $A \cup B$ lengoaia erregularra da
 - IV. $A \cdot B$ lengoaia erregularra da
 - V. A^* lengoaia erregularra da
 - VI. Beste ezer ez da Σ gaineko lengoaia erregularra

Adierazpen erregularrak

- Bedi Σ alfabetoa:
 ϵ adierazpen erregularra da.
$$L(\epsilon) = \{\epsilon\}$$
 - II. $a \in \Sigma$ alfabetoko ikur bakoitzerako, adierazpen erregularra da.
$$L(a) = \{a\}$$
- Bitez e_1 eta $e_2 \in \Sigma$ gaineko bi adierazpen erregular:
 - III. $(e_1) \mid (e_2)$ adierazpen erregularra da.
$$L((e_1) \mid (e_2)) = L(e_1) \cup L(e_2)$$
 - IV. $(e_1) \cdot (e_2)$ adierazpen erregularra da.
$$L((e_1) \cdot (e_2)) = L(e_1) \cdot L(e_2)$$
 - V. $(e_1)^*$ adierazpen erregularra da.
$$L((e_1)^*) = L(e_1)^*$$
 - VI. Beste ezer ez da Σ gaineko adierazpen erregularra

Definizio erregularrak

- Adierazpen erregularren idazketa errazteko notazioa

$$\text{def}_1 \rightarrow \text{ae}_1$$
$$\text{def}_2 \rightarrow \text{ae}_2$$
$$\dots$$

non def_i identifikadoreak eta ae_i def_i gehiago izan ditzaketen adierazpen erregularrak diren

Espezifikazioaren adibidea (1. hurbilpena)

Token mota	Deskribapena	Atributuak	Adib.
Osoko konstantea	(0 1 2 3 4 5 6 7 8 9)+	Izena: string Balio: osoko	00 1245
...

Automata finituak

- » A automata batek L ezagutzen du:
 - Lko hitzak (kateak) onartzen ditu
 - gainontzekoak baztertu egiten ditu
- » Eredu ezagutzaile baten elementuak:
 - a) Sarrera zinta → alfabetoko ikurren segida
 - b) Irakurketa burua → sarrerako elementu bat erakusten du
 - c) Kontrol finitua → trantsizio funtzio batek zuzendua, honek definitzen du ezagutu beharreko lengoaia

Automata finituak

- Programa ezagutzaile bat lortzeko prozesua:
 - a) lengoaiarako adierazpen erregularra definitu
 - b) trantsizio funtzioa lortu (AFD)
 - c) programa ezagutzailea idatzi
- Automata finituak:
 - » deterministak (AFD)
 - » ez deterministak (AFE)

Automata finituak

- AFD: $A = (Q, \Sigma, \delta, q_0, F)$ non
 - Q : egoera multzoa
 - Σ : alfabetoa
 - $\delta : Q \times \Sigma \rightarrow Q$
 - q_0 : hasierako egoera
 - F : bukaerako egoeren multzoa

Automata finituak

- AFD baten errepresentazioa: trantsizioen grafoa
- Konfigurazioa: $(q, w\#)$ egoera eta irakurtzeke dagoena
- Mugimendua:
 - $(q, aw\#) \vdash (q_1, w\#)$ baldin $q_1 = \delta(q, a)$
- Onartutako hitza: $(q_0, w\#) \vdash^* (q_f, \#)$ q_f barne F
- AFD: egoera eta ikur bakoitzeko trantsizio bakarra gehienez

Automata finituak

- ε -AFE: $A = (Q, \Sigma, \delta, q_0, F)$ non
 - Q : egoera multzoa
 - Σ : alfabetoa
 - $\delta : Q \times \Sigma \cup \{\varepsilon\} \rightarrow \mathcal{P}(Q)$ egoera multzoa
 - q_0 : hasierako egoera
 - F : bukaerako egoeren multzoa

Automata finituak

- ε -AFE baten errepresentazioa: trantsizioen grafoa
- Konfigurazioa: $(q, w\#)$ egoera eta irakurtzeke dagoena
- Mugimendua:
 - $(q, aw\#) \vdash (q_1, w\#)$ baldin q_1 barne $\delta(q, a)$
 - $(q, w\#) \vdash (q_1, w\#)$ baldin q_1 barne $\delta(q, \varepsilon)$
- Onartutako hitza: $(q_0, w\#) \vdash^* (q_f, \#)$ q_f barne F
- AFE: egoera eta ikur bakoitzeko trantsizio multzo bat

Baliokidetza eta bihurketa

- Erabaki al dezakegu $x \in L(\alpha)$ den, non α AE bat den?
- Eskuz idatz genezake $x \in L(\alpha)$ den erabakiko duen automata bat.
- Zorionez badira baliokidetasuna bermatzen duten bihurketa algoritmoak
 - » $AE \Rightarrow \varepsilon\text{-AFE} \Rightarrow \text{AFD}$

2.3 Analizatzaile baten diseinu eta inplementazioa

- Automata finitu determinista batean oinarrituta eraiki: `hurrengo_tokena()`

Inplementazioari buruz:

- AFD bakarra (??)
- Buffer-en erabilera
- Hitz gakoan (erreserbatuak) tratamendua
- Trantsizio taularen inplementazio eraginkorra
- Egoera finituei lotutako ekintzak

AFD baten algoritmoa

- *Edozein A automatarako*

q:=hasierako_egoera(A) ;

irakurri c-en lehenbiziko karakterea;

bitartean trantsizioa_dago(A,q,c) egin

q:=trantsizioa(A,q,c);

c:=hurrengo karakterea;

ambitartean;

baldin bukaerakoa(A,q) orduan itzuli “bai”

bestela itzuli “ez”

Hurrengo_tokena

*{c-ek sarrera fitxategiko indarrean dagoen
karakterea dauka}*

errepika

lortu_tokena (t);

harik eta

t analizatzaile sintaktikoarentzat
interesgarria den token bat izan arte

amerrepika

lortu_tokena

Indarrean dagoen karakterean oinarrituta tokena lortu

{c-ek sarrera fitxategiko indarrean dagoen karakterea dauka}

q:=hasierako_egoera(A) ;

bitartean trantsizioa_dago(A,q,c) egin

q:=trantsizioa(A,q,c);

c:=hurrengo karakterea;

-- fitxategi bukaeran bagaude ez du “kaskatzen”

ambitartean;

baldin bukaerakoa(A,q) orduan q egoerari dagokion tokena itzuli

bestela itzuli “okerreko tokena”

Automata datu-mota

- **Automata sortzeko eragiketak**

- » Sortu
- » Egoera_gehitu
- » Ikurra_gehitu
- » Trantsizioa_gehitu
- » Definitu_hasierako_egoera
- » Definitu_bukaerako_egoera

Automata datu-mota

- **Bestelako eragiketak**

- » `trantsizioa`: `Automata × egoera × ikurra → egoera`
- » `trantsizioa_dago`: `Automata × egoera × ikurra → boolearra`
- » `hasierako_egoera`: `Automata → egoera`
- » `bukaerakoa`: `Automata × egoera → boolearra`

- **Automata datu-motarako errepresentazio posiblea:**

- » Lau osagaidun erregistroa (hasierako egoera, egoerak, bukaerako egoerak, trantsizioak).
- » Trantsizioak bi dimentsiotako taula baten bidez (egoerak, ikur-multzoak).

Bestelakoak (I)

- AFD bakoitzeko bat?
- Guztiak bakar batean bildu?
- Karakterez karaktere irakurtzea ez-eraginkorra izan daiteke:
 - » Lerroz lerro: kontuz lerro amaierarekin. Lerro anitzeko iruzkinak?
 - » Bufferra erabili

Bestelakoak (II)

- Hitz erreserbatuak:
 - » Automata bat hitz erreserbatu bakoitzeko
 - » Salbuespena egin: hitz_erres(str) funtzioa
- Trantsizio taula:
 - » 256 ASCII karaktere daude \Rightarrow 256 zutabe
 - » Zutabe asko berdin-berdinak dira:
karakte multzo batez etiketatutako zutabea
(datu-mota berria)

Bestelakoak (III)

- Egoerekin lotutako ekintzak:
 - » Automatak ez-bukaerako egoera batean amaitzen badu: **ERROREA**
 - » Bukaerako egoera batean amaitzen badu:
 - Lotutako ekintzak egikaritu
 - Dagokion token mota itzuli
 - » Implementazio aukera: case q

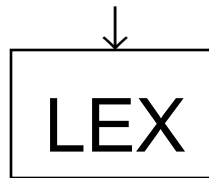
Bestelakoak (IV)

Ekintzak: an. lex. espezifikazioa

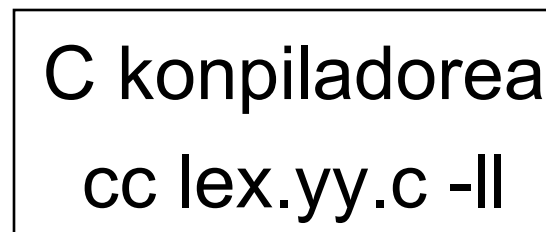
Token mota	Deskribapena	Atributuak	Ekintzak
ident	$l(l d)^*$	izena: S_T_erref	baldin hitzerres(str) orduan mota:=HITZERRES; izena:=bilatu_HE(str); bestela mota:=IDENT; str:=normalizatu(str); izena:=gehitu(S_T,str);
hitz. erres.	program var ...	izena: enumeratu	
kte_osoko	d^+	balio: osoko	mota:=KTEOSOKO; balio:=str2int(str);
kte_erreal	$d^+.d^+$	balio: erreal	mota:=KTEERREAL; balio:=str2real(str);

2.4 Analizatzaile lexikoen espezifikaziorako lengoaia: LEX

LEX espezifikazioa



lex.yy.c yylex()



a.out

Analizatzaile lexikoen espezifikaziorako lengoaia: LEX

Sarrera
katea



Token
segida

LEX espezifikazioak (I)

ERAZAGUPENAK

%%

ERREGELAK

%%

ERABILTZAILEAREN AZPIERRUTINAK

LEX espezifikazioak(II)

ERAZAGUPENAK

Aldagai/konstanteak

Definizio erregularrak

ERREGELAK

pi {i-garren ekintza}

LEX espezifikazioak(III)

ERABILTZAILEAREN AZPIERRUTINAK

Prozedura lagungarriak

LEX-ek erabiltzen dituen prozeduren
birdefinizioa

Adierazpen erregularrak LEX-en (I)

- Testu-karakterek / Karaktere bereziak
- ESCAPE \ eta komatxoak "
- PUNTUA. Lerro bukaera ez den edozein karaktere
- KARAKTERE MULTZOAK
 - » [] [AB] A eta B ikurrak dituen multzoa
 - » - heina [A-Z] Atik Zrako ikur guztiak dituen
 - Heina multzoaren barruan soilik erabil daiteke
 - » ^ Osagarria [^abc] a, b edo c ikurra ez direnen multzoa
 - Multzo baten osagarria da, beraz multzoaren barruan
 - » \ Escape \n lerro jauzia \t tabulazioa
\[edo "[" kortxetea \a edo "a" a ikurra

Adierazpen erregularrak LEX-en (II)

- AUKERAKO ADIERAZPENA

? ab?c -> abc ac

- ADIERAZPEN ERREPIKATUAK * +

- AUKERAK | a|b a ala b ikurra

- TESTUINGURUA:

» ^ \$

^ -k lerro hasiera adierazten du eta

\$ -ek lerro bukaera adierazten du:

`^[A-Z].*[A-Z]$` Maiuskulaz hasi eta amaitzen den lerroa

- Hitz hutsa ez dago LEX-en

LEX ekintzak

- Ez badago →kopiatu
 - » ez dagoenean parekatzerik
- EKINTZA HUTSA ;
- ECHO
- yymore()
- ...
- yywrap()

LEX-ek erabiltzaileari eskaintzen dizkion egiturak

- **yytext**

- » karaktereen array-a

- **yylen**

- » parekatutako karaktere kopurua

Anbiguotasunaren tratamendua

Parekatze luzeena



LEX definizioan dagoen lehenbiziko ekintza

- Adibidez: if {1 ekintza}
 [a-z]+ {2 ekintza}

LEX

- Erregelak kode bihurtzen ditu
- Zuriunez hasten den edozein lerro `lex.yy.c` programara kopiatzen du
 - » LEX-etik kanpoko aldagaien erazagupena
 - » LEX mailan dauden aldagaien erazagupena
- `%{` eta `%}` artean dagoen LEX sarrera ere dagoen bezala kopiatzen du

Adibidez

%%

[bv] {printf("b edo v irakurri dut ");}

[^b] {ezb();}

. ;

%%

ezb() {printf("<> b");}