Interpretacija programa
Sintaksni analizator

Krunoslav Puljić

Podatkovna struktura SA

- Podatkovnu strukturu SA čine:
  - Globalni podaci
    - Tablica znakova i stog
  - Lokalni podaci
    - Koristi ih isključivo SA za potrebe izvođenja izabranog procesa parsiranja
- Tablica uniformnih znakova
  - Sadrži niz uniformnih znakova zapisanih onim redoslijedom kojim se leksičke jedinke pojavljuju u izvornom programu
  - Osnovna i najznačajnija struktura
  - Upravo sintaksna pravila propisuju na koji način se grade ispravne sintaksne cjeline leksičkih jedinki

Podatkovna struktura SA

- Tablica ključnih riječi, operatora i specijalnih znakova
  - Druga po važnosti tablica
  - Npr. koriste se podaci o prioritetima operatora, da li je operator prekidni znak...
- Tablica identifikatora
  - Rijetko se koristi
  - Npr. provjera indeksa u polju ponekad se smatra sintaksnim, a ponekad sematičkim pravilom
    - Npr. podaci o dozvoljenom rasponu indeksa unutar polja zapisuju se upravo u tablicu identifikatora

Podatkovna struktura SA

- Stog
  - Većina sintaksnih i semantičkih analizatora razmjenjuje podatke primjenom stoga
    - Bez obzira na primijenjenu tehniku programskog ostvarenja
  - Gradi se izravno za potrebe procesa parsiranja
    - Ili se koristi stog jezika izgradnje jezičnog procesora u slučaju primjene tehnika rekurzivnog spusta

Podatkovna struktura SA

- Sintaksni analizator:
  - Grupira uniformne znakove u sintaksne cjeline
  - Stvara hijerarhijsku strukturu sintaksnih cjelina
  - Stavlja sintaksne cjeline na stog
  - Predaje stog semantičkom analizatoru

Podatkovna struktura SA

- Lokalna struktura podataka
  - Gradi se za potrebe procesa parsiranja
  - Ovisi o primijenjenoj tehnici parsiranja
    - Tehnike parsiranja od vrha prema dnu koriste tablicu u kojoj su spremljene sve produkcije gramatike
    - Tehnike parsiranja od dna prema vrhu koriste tablice u kojima se definiraju akcije parsera, kao i tablicu redukcija
    - Posebne metode parsiranja koriste tablice u kojima se definiraju dopušteni parovi susjednih operatora
      - npr. parsiranje primjenom CoNo tablica – Current operator, Next Operator

Opis sintaksnih pravila

- Sintaksna pravila najčešće se definiraju primjenom:
  - konteksno neovisne gramatike
  - regularnih izraza
  - BNF sustava oznaka
  - sustava oznaka COBOL

BNF sustav oznaka

- BNF = Backus-Naur Form
  - Ili Backus Normal Form
- Sličan je sustavu oznaka konteksno neovisne gramatike
- Predložen još 1960.g. za potrebe opisa prigramskog jezika IAL (kasnije ALGOL)
- Osnovni elementi jezika definiraju se primjenom pravila (produkcije u KNG)

BNF sustav oznaka

- Osnovni elementi jezika definiraju se primjenom:
  - Pravila
    - produkcije u KNG
  - Izraza
  - Konstanti
    - znakovi jezika - završni znakovi u KNG
  - Varijabli
    - Sintaksne cjeline - nezavršni znakovi u KNG
  - Znaka jednakosti
    - Znak koji dijeli lijevu i desnu stranu produkcije u KNG
  - Operatora nadovezivanja
  - Operatora izbora

BNF sustav oznaka

- Dozvoljava rekurzivne definicije
- Opis jezika zadaje se nizom pravila
  - Pravila imaju lijevu i desnu stranu odvojenu znakom jednakosti
  - Lijevu stranu čini točno jedna varijabla
  - Desnu stranu čini više izraza odvojenih operatorom izbora
  - Izraz je niz varijabli i konstanti
  - Dozvoljava se da desna strana nema niti jedan izraz ili da ima znak ε

BNF sustav oznaka

- 1960.godine Naur koristi sljedeći sustav oznaka:
  - ::= znak jednakosti
  - < v > varijabla
  - znak_do_znaka nadovezivanje
  - | operator izbora
- Primjer: cijeli brojevi se definiraju kao:
  - <CijeliBroj>::=<Brojka>|<CijeliBroj><Brojka>
  - <Brojka>::=0|1|2|3|4|5|6|7|8|9
    - 0...9 su konstante, <Brojka> je varijabla
    - Definicija cijelog broja je rekurzivna

BNF sustav oznaka

- BNF sustavom oznaka moguće je opisati njega samog
  - Neka su VARIJABLA i KONSTANTA konstante
    - Označavaju varijable i konstante koje se već koriste u jeziku
  - Da se izbjegne nejednoznačnost korištenja znakova | i ::=, definiraju se “|” i “::=”
    - Jer su | i ::= istodobno i operatori sustava i oznaka i konstante

BNF sustav oznaka

- BNF definiramo ovako:
  - <BNF> ::= <BNF><Pravilo> | <Pravilo>
  - <Pravilo> ::= <Pravilo> “|” <Izraz> |
    VARIJABLA ”::=“ <Izraz> |
    VARIJABLA ”::=“
  - <Izraz> ::= <Izraz> <Znak> | <Znak>
  - <Znak> ::= VARIJABLA | KONSTANTA

SA zasnovana na Co-No tablici

- Co-No = Current Operator – Next Operator
- Jedan od najstarijih načina analize i prevođenja
- Osnovna prednost je brzina
- Nedostatak je neučinkovito korištenje memorijskog prostora

SA zasnovana na Co-No tablici

- Primjer:
  - Neka se naredbe ulančavaju operatorom ;
  - U naredbama je moguće zadati aritmetičke izraze koji imaju najviše dva operanda
  - Koriste se operacije +, -, * i /
  - Znak pridruživanja je $\rightarrow$
- Neka je zadan niz naredbi:

```
;
0 → Sljedeci;
Sljedeci + 1 → Sljedeci;
Porast * Sljedeci → Umnozak;
```

SA zasnovana na Co-No tablici

- Ispravnost izvornog programa i prevođenje u strojni program temelji se na poznavanju dva podatka:
  - Lijevog i desnog operatora
- Neka ciljni program koristi samo jedan registar procesora, npr. D0
- Analiza započinje operatorima ; i →

```
;
0 → Sljedeci;
Sljedeci + 1 → Sljedeci;
Porast * Sljedeci → Umnozak;
```

SA zasnovana na Co-No tablici

- Operator ; je lijevi operator, a → je desni

```
; 0 →
```

- Definira se da je promatrani par operatora dozvoljen i da se za taj par operatora generira naredba strojnog jezika koja sprema vrijednost 0 u registar D0

```
;
0 → Sljedeci;
Sljedeci + 1 → Sljedeci;
Porast * Sljedeci → Umnozak;
```

SA zasnovana na Co-No tablici

- Idući par operatora je → i ;

```
→ Sljedeci;
```

- Definira se da je promatrani par operatora dozvoljen i da se za taj par operatora generira naredba strojnog jezika koja sprema vrijednost iz registra D0 na memorijsku lokaciju određenu varijablom Sljedeci

```
→ Sljedeci;
Sljedeci + 1 → Sljedeci;
Porast * Sljedeci → Umnozak;
```

SA zasnovana na Co-No tablici

- Idući par operatora je $\rightarrow$ i ;

```
; Sljedeci +
```

- Definira se da je promatrani par operatora dozvoljen i da se za taj par operatora generira naredba strojnog jezika koja dohvaća vrijednost memorijske lokaciju određene varijablom Sljedeci i sprema ju u registar D0

```
;
Sljedeci + 1 → Sljedeci;
Porast * Sljedeci → Umnozak;
```

SA zasnovana na Co-No tablici

- Idući par operatora je → i ;

```
+ 1 →
```

- Definira se da je promatrani par operatora dozvoljen i da se za taj par operatora generira naredba strojnog jezika koja dodaje vrijednost 1 u registar D0

```
+ 1 → Sljedeci;
Porast * Sljedeci →
Umnozak;
```

SA zasnovana na Co-No tablici

- Idući par operatora je → i ;

```
    → Sljedeci;
```

- Definira se da je promatrani par operatora dozvoljen i da se za taj par operatora generira naredba strojnog jezika koja sprema sadržaj registra D0 na memorijsku lokaciju određenu varijablom Sljedeci

```
    → Sljedeci;
    Porast * Sljedeci →
    Umnozak;
```

- itd.

SA zasnovana na Co-No tablici

- Cijeli postupak analize izvornog programa i generiranja ciljnog programa zadaje se 2D tablicom veličine N×N, gdje je N broj različitih operatora koji se koriste u izvornom programu
  - Neka redovi tablice označavaju lijevi operator, a stupci desni (Co-No tablica)
    - Ako je par operatora dozvoljen sintaksnim pravilima, onda element tablice određen navedenim parom označava jednu od akcija generatora ciljnog programa
    - Inače, u element tablice zapisujemo oznaku pogreške

SA zasnovana na Co-No tablici

	;	→	+	-	×	/
;	*Greška*	**dohvati**	**dohvati**	**dohvati**	**dohvati**	**dohvati**
→	**spremi**	*Greška*	*Greška*	*Greška*	*Greška*	*Greška*
+	*Greška*	**zbroji**	*Greška*	*Greška*	*Greška*	*Greška*
-	*Greška*	**oduzmi**	*Greška*	*Greška*	*Greška*	*Greška*
×	*Greška*	**pomnozi**	*Greška*	*Greška*	*Greška*	*Greška*
/	*Greška*	**podijeli**	*Greška*	*Greška*	*Greška*	*Greška*

SA zasnovana na Co-No tablici

- Većinu prethodne tablice čine oznake pogreške
- Neka su zadane sljedeće naredbe:

```
;
0 → Sljedeci;
Sljedeci + 1 → Sljedeci;
Porast * Sljedeci – 100 → Umnozak;
```

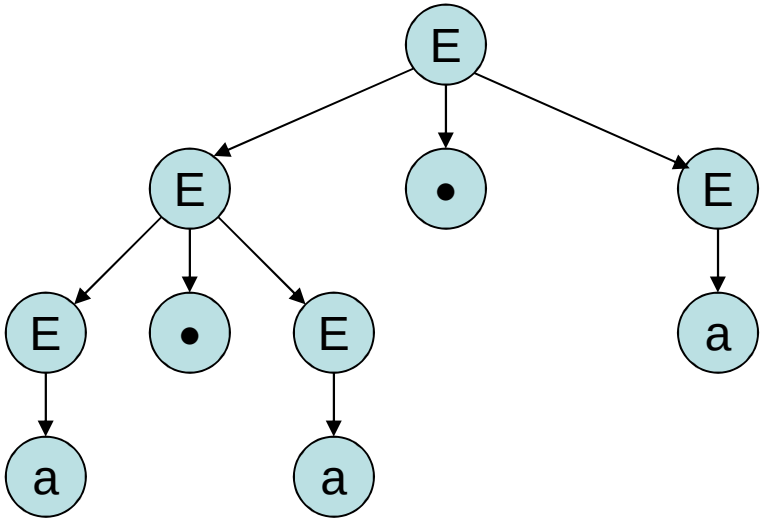
- Na temelju zadane Co-No tablice, SA pronalazi pogrešku u posljednjoj liniji tijekom analize para operatora × i –
  - Navedeni par nije dozvoljen

Nejednoznačnost

- Interpretacija niza temelji se na generativnom stablu koje se gradi tijekom generiranja niza
- Mogućnost izgradnje više različitih stabala uzrokuje nejednoznačnost u interpretaciji niza

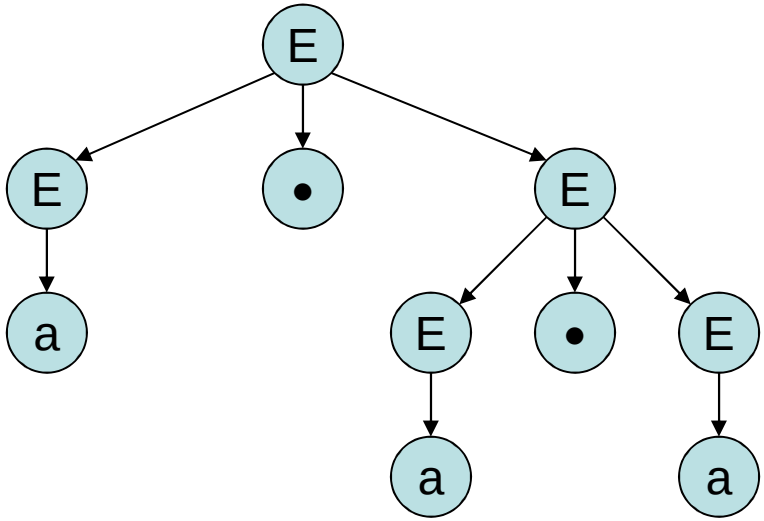
Nejednoznačnost

- Primjer: neka je zadana gramatika G=({E}, {a,•}, {E→E•E|a}, E)
- Za niz a•a•a moguće je izgraditi dva generativna stabla:
  - $\underline{E} \rightarrow \underline{E}\bullet E \rightarrow \underline{E}\bullet E\bullet E \rightarrow a\bullet\underline{E}\bullet E \rightarrow a\bullet a\bullet\underline{E} \rightarrow a\bullet a\bullet a$



Nejednoznačnost

- Primjer: neka je zadana gramatika G=({E}, {a,•}, {E→E•E|a}, E)
- Za niz a•a•a moguće je izgraditi dva generativna stabla:
  - $\underline{E} \rightarrow \underline{E}\bullet E \rightarrow a\bullet\underline{E} \rightarrow a\bullet\underline{E}\bullet E \rightarrow a\bullet a\bullet\underline{E} \rightarrow a\bullet a\bullet a$



Nejednoznačnost

- Primjer: neka je zadana gramatika G=({E}, {a,•}, {E→E•E|a}, E)
- Primjenom produkcija istim redoslijedom kao u drugom primjeru, ali ne na iste nezavršne znakove, dobije se stablo iz prvog primjera
  - $\underline{E} \rightarrow \underline{E}•E \rightarrow a•\underline{E} \rightarrow a•\underline{E}•E \rightarrow a•a•\underline{E} \rightarrow a•a•a$
  - $\underline{E} \rightarrow E•\underline{E} \rightarrow \underline{E}•a \rightarrow E•\underline{E}•a \rightarrow \underline{E}•a•a \rightarrow a•a•a$
- Grupiranje završnih znakova određuje redoslijed izvođenja operacija

Nejednoznačnost

- Generativno stablo određuje interpretaciju niza
  - Npr. neka u prethodnom primjeru završni znak a označava varijablu ili konstantu, a • operator
    - Generativno stablo tada definira prioritet operatora
    - Ako uvedemo zagrade, prvo stablo označava (a•a)•a, a drugo a•(a•a)
- Grupiranje završnih znakova određuje redoslijed izvođenja operacija
  - Ako je •
    - operator +, onda je (a1+a2)+a3 = a1+(a2+a3)
    - operator –, onda je (a1–a2)–a3 ≠ a1–(a2–a3)

Nejednoznačnost

- Zbog svega nevedenog generativno stablo ima važnu ulogu u interpretaciji niza
  - Nastoji se konstruirati gramatika koja za zadani niz gradi samo jedno generativno stablo
- Na temelju primjera zaključujemo:
  - Za bilo koji niz KNG moguće je izgraditi jedno ili više različitih generativnih stabala
  - Bilo koje generativno stablo moguće je izgraditi primjenom jednog ili više različitih postupaka generiranja niza

Generiranje niza

- U praksi se koriste:
  - Generiranje niza zamjenom krajnje lijevog nezavršnog znaka
  - Generiranje niza zamjenom krajnje desnog nezavršnog znaka

Generiranje niza zamjenom krajnje lijevog nezavršnog znaka

- Postupak generiranja niza u kojemu se produkcije primjenjuju isključivo na krajnje lijeve nezavršne znakove u međunizu
- Npr.

$\underline{E} \rightarrow$		$\underline{E} \rightarrow$
$\underline{E} \bullet E \rightarrow$		$\underline{E} \bullet E \rightarrow$
$\underline{E} \bullet E \bullet E \rightarrow$	ili	$a \bullet \underline{E} \rightarrow$
$a \bullet \underline{E} \bullet E \rightarrow$		$a \bullet \underline{E} \bullet E \rightarrow$
$a \bullet a \bullet \underline{E} \rightarrow$		$a \bullet a \bullet \underline{E} \rightarrow$
$a \bullet a \bullet a$		$a \bullet a \bullet a$

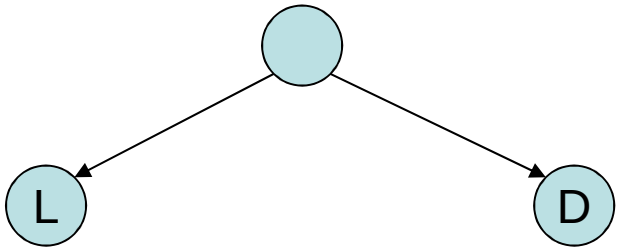
Generiranje niza zamjenom krajnje desnog nezavršnog znaka

- Postupak generiranja niza u kojemu se produkcije primjenjuju isključivo na krajnje desne nezavršne znakove u međunizu
- Npr.

$\underline{E} \rightarrow$		$\underline{E} \rightarrow$
$E \bullet \underline{E} \rightarrow$		$E \bullet \underline{E} \rightarrow$
$\underline{E} \bullet a \rightarrow$	ili	$E \bullet E \bullet \underline{E} \rightarrow$
$E \bullet \underline{E} \bullet a \rightarrow$		$E \bullet \underline{E} \bullet a \rightarrow$
$\underline{E} \bullet a \bullet a \rightarrow$		$\underline{E} \bullet a \bullet a \rightarrow$
$a \bullet a \bullet a$		$a \bullet a \bullet a$

Obilazak generativnog stabla

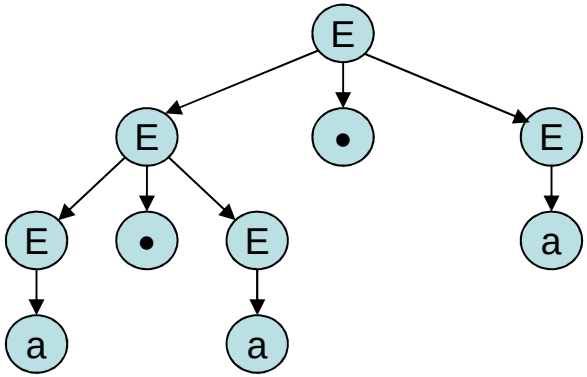
- Postupak obilaska generativnog stabla određuje redoslijed kojim se obilaze grane i čvorovi
- Uobičajeni postupci su:
  - Lijevi obilazak
  - Desni obilazak



- Postupci obilaska jednoznačno definiraju redoslijed primjene produkcija i redoslijed nezavršnih znakova na koje se produkcije primjenjuju

Obilazak generativnog stabla

- Npr. lijevi obilazak stabla definira
  - $\underline{E} \rightarrow \underline{E}\bullet E \rightarrow \underline{E}\bullet E\bullet E \rightarrow a\bullet\underline{E}\bullet E \rightarrow a\bullet a\bullet\underline{E} \rightarrow a\bullet a\bullet a$
  - Zamjena krajnje lijevog nezavršnog znaka
- Npr. desni obilazak stabla definira
  - $\underline{E} \rightarrow E\bullet\underline{E} \rightarrow \underline{E}\bullet a \rightarrow E\bullet\underline{E}\bullet a \rightarrow \underline{E}\bullet a\bullet a \rightarrow a\bullet a\bullet a$
  - Zamjena krajnje desnog nezavršnog znaka



Generativno stablo

- Tvrdnje:
  - Bilo koje generativno stablo moguće je izgraditi primjenom jednog i samo jednog postupka generiranja niza zamjenom krajnje lijevog nezavršnog znaka
  - Bilo koje generativno stablo moguće je izgraditi primjenom jednog i samo jednog postupka generiranja niza zamjenom krajnje desnog nezavršnog znaka

Nejednoznačnost

- KNG G je nejednoznačna ako postoji niz $w \in L(G)$ za koji je moguće izgraditi više različitih generativnih stabala
  - Tvrdnja: ako postoji niz $w \in L(G)$ koji je moguće generirati primjenom više različitih postupaka generiranja niza zamjenom krajnje desnog/lijevog nezavršnog znaka, onda je gramatika G nejednoznačna
- Npr. gramatika $G=(\{E\}, \{a,\bullet\}, \{E \rightarrow E \bullet E | a\}, E)$ jest nejednoznačna

Nejednoznačnost

- Niz $w \in L(G)$ za koji je moguće izgraditi više različitih generativnih stabala jest nejednoznačan niz za gramatiku G
  - Npr. niz a•a•a jest nejednoznačan niz za gramatiku $G=(\{E\}, \{a,\bullet\}, \{E \rightarrow E \bullet E | a\}, E)$
- Jezik L je inherentno nejednoznačan ako ga nije moguće generirati niti jednom jednoznačnom gramatikom

Nejednoznačnost

- Jezik L kojega generira gramatika G=({E}, {a,•}, {E→E•E|a}, E) moguće je generirati primjenom više različitih jednoznačnih gramatika
  - ... različitih i od gramatike G
- Za lijevo asocijativni operator • gradi se sljedeća jednoznačna gramatika G1

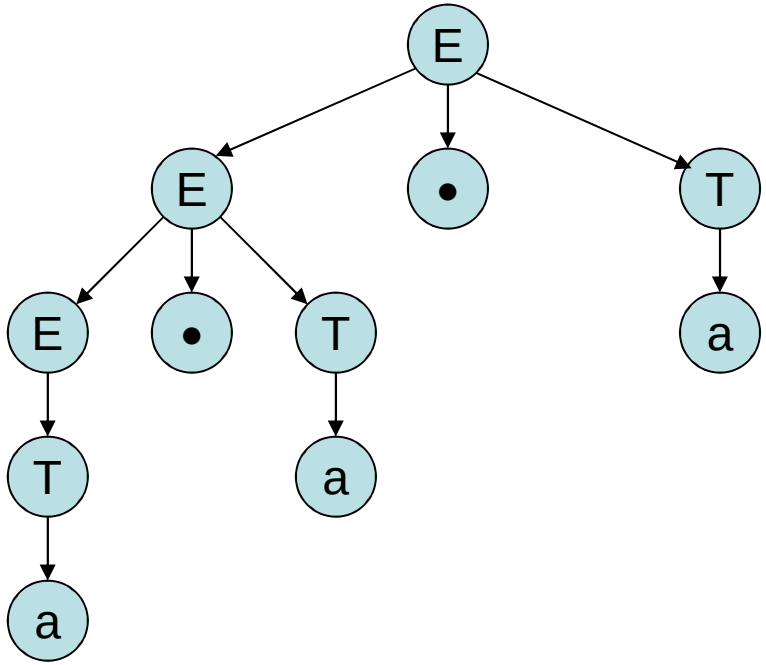
  G1= ({E,T}, {a,•}, {E→E•T|T, T→a}, E)

Nejednoznačnost

- G1= ({E,T}, {a,•}, {E→E•T|T, T→a}, E)
- Niz a•a•a je jednoznačan za gramatiku G1, jer ga je moguće generirati:
  - samo jednim postupkom generiranja niza zamjenom krajnje lijevog nezavršnog znaka
    - $\underline{E} \rightarrow \underline{E}\bullet T \rightarrow \underline{E}\bullet T\bullet T \rightarrow \underline{T}\bullet T\bullet T \rightarrow a\bullet\underline{T}\bullet T \rightarrow a\bullet a\bullet\underline{T} \rightarrow a\bullet a\bullet a$
  - samo jednim postupkom generiranja niza zamjenom krajnje desnog nezavršnog znaka
    - $\underline{E} \rightarrow E\bullet\underline{T} \rightarrow \underline{E}\bullet a \rightarrow E\bullet\underline{T}\bullet a \rightarrow \underline{E}\bullet a\bullet a \rightarrow \underline{T}\bullet a\bullet a \rightarrow a\bullet a\bullet a$

Nejednoznačnost

- G1= ({E,T}, {a,•}, {E→E•T|T, T→a}, E)
- Generativno stablo niza a•a•a



Parsiranje niza

- Osim primjene gramatike za generiranje jezika, gramatike koristimo i za određivanje pripadnosti proizvoljnog niza w jeziku L(G)
- Proces određivanja pripadnosti niza w jeziku L(G) naziva se prepoznavanje niza
  - Ako se ustanovi da w pripada jeziku L(G), potrebno je izgraditi generativno stablo koje će se koristiti za interpretaciju niza
- Proces prepoznavanja niza i izgradnje generativnog stabla naziva se parsiranje niza

Parsiranje niza

- Generativno stablo se često naziva i stablo parsiranja
- Po načinu gradnje generativnog stabla razlikujemo dvije metode parsiranja:
  - Parsiranje od vrha prema dnu
    - Gradi stablo od korijena stabla (početnog nezavršnog znaka gramatike) prema listovima stabla (završnim znakovima gramatike)
  - Parsiranje od dna prema vrhu
    - obrnuto

Parsiranje od vrha prema dnu

- Zadana je gramatika G=(V,T,P,S)
- Parsiranje od vrha prema dnu
  - započinje gradnju stabla od početnog nezavršnog znaka S
  - Ostali čvorovi grade se primjenom produkcija iz skupa P gramatike G
  - Produkcije se primijenjuju sve dok se listovi stabla ne označe isključivo završnim znakovima zadanog niza $w \in T^*$
  - Tijekom gradnje stabla završni znakovi niza w određuju koja se produkcija primijenjuje

Parsiranje od vrha prema dnu

- Široka primjena
  - Jednostavno programski ostvarivo
  - Učinkovito

Parsiranje od vrha prema dnu

- Primjer: gramatika koja generira begin-end blokove programskog jezika (npr.Pascal)
  - Nezavršni znakovi V={C, S, S1, S2}
  - Početni nezavršni znak je C
  - Skup završnih znakova

    T = {begin, end, while, do, ;, :=, ≠, var}
    - Znakovi begin, end, while, do su ključne riječi jezika
    - var je varijabla
    - Napomena: jedan završni znak označen je kao niz slova

Parsiranje od vrha prema dnu

- Primjer: gramatika koja generira begin-end blokove programskog jezika (npr.Pascal)
  - Skup završnih znakova

    T = {begin, end, while, do, ;, :=, ≠, var}
    - := je operator pridruživanja
    - ≠ je operator nejednakosti
    - ; je oznaka ulančavanja naredbi

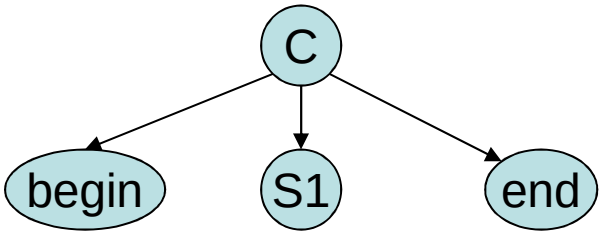
Parsiranje od vrha prema dnu

- Primjer: gramatika koja generira begin-end blokove programskog jezika (npr.Pascal)
  - Skup produkcija P je:
    - C $\rightarrow$ begin S1 end
    - S1 $\rightarrow$ S S2
    - S2 $\rightarrow$; S1 | ε
    - S $\rightarrow$ var := var |
      while var $\neq$ var do S |
      begin S1 end

Parsiranje od vrha prema dnu

- Provjerimo da li sljedeći niz pripada jeziku

```
begin     var := var;
   while var ≠ var do
          var := var
end
```



- Parsiranje počinje od korijena stabla, tj. od početnog nezavršnog znaka C
  - Na početku je moguće primjeniti samo produkciju C→begin S1 end
    - To je jedina produkcija s C na lijevoj strani
    - Zato bilo koji niz jezika mora započeti znakom begin, pa nastavljamo parsiranje

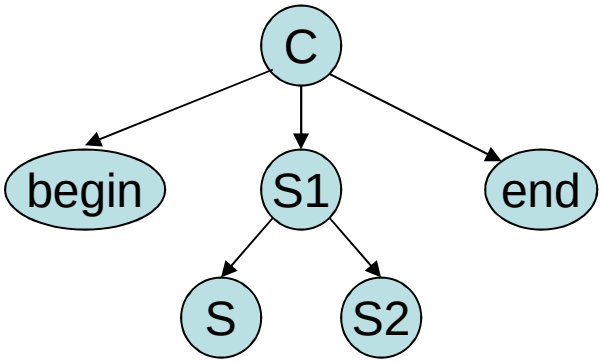
```
begin S1
end
```

Parsiranje od vrha prema dnu

- Provjerimo da li sljedeći niz pripada jeziku

```
begin     var := var;
   while var ≠ var do
             var := var
end
```

```
begin     S S2
end
```



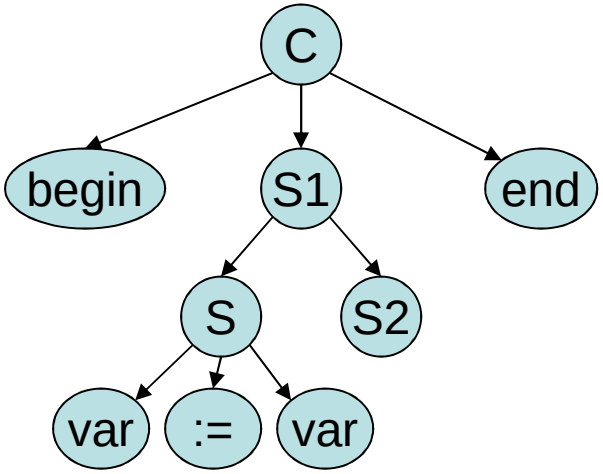
- Slijedi produkcija S1→S S2
  - To je jedina produkcija s S1 na lijevoj strani
  - Nastavljamo parsiranje

Parsiranje od vrha prema dnu

- Provjerimo da li sljedeći niz pripada jeziku

```
begin     var := var;
   while var ≠ var do
          var := var
end
```

```
begin     var := var S2
end
```



- Parsiranje se nastavlja zamjenom krajnje lijevog nezavršnog znaka S
  - Za S postoje 3 produkcije
  - Budući da se iza niza begin nalazi niz var primijenjujemo produkciju S→var:=var

Parsiranje od vrha prema dnu

- ltd.
- U primjeru nismo imali mogućnost izbora više produkcija
  - Ako postoji takva mogućnost, potrebno je ispitati sve moguće kombinacije primjena produkcija
- Opisana gramatika i proces parsiranja od vrha prema dnu naziva se LL(1) gramatika i LL(1) parsiranje
  - Prvi L u LL označava da se ulazni niz čita s lijeva na desno (Left-to-right scanning), a drugi L označava da se produkcije primjenjuju na krajnje lijevi nezavršni znak u generiranom međunizu (Leftmost derivation)
  - (1) znači da se gleda 1 token unaprijed

LL(1) i LR(1)

- U praksi se često koriste LR(1) gramatike
  - Prvi L je čitanje s lijeva na desno
  - Drugi R je znači da se produkcije primjenjuju na krajnje desni nezavršni znak
  - (1) znači da se gleda 1 token unazad/unaprijed

YACC

- Regularni izrazi – LEX
- Gramatike – YACC
  - Zadatak: prođite kroz primjer na
    - http://ds9a.nl/lex-yacc/cvs/lex-yacc-howto.html
    - http://courses.washington.edu/cssap442/zander/yacc.html
    - http://dinosaur.compilertools.net/yacc/
    - http://publib.boulder.ibm.com/infocenter/aix/v6r1/index.jsp?topic=/com.ibm.aix.genprogc/doc/genprogc/ie_prog_4lex_yacc.htm

Literatura

- Siniša Srbljić: Jezični procesori 1 **[JP1]**
- Siniša Srbljić: Jezični procesori 2 **[JP2]**