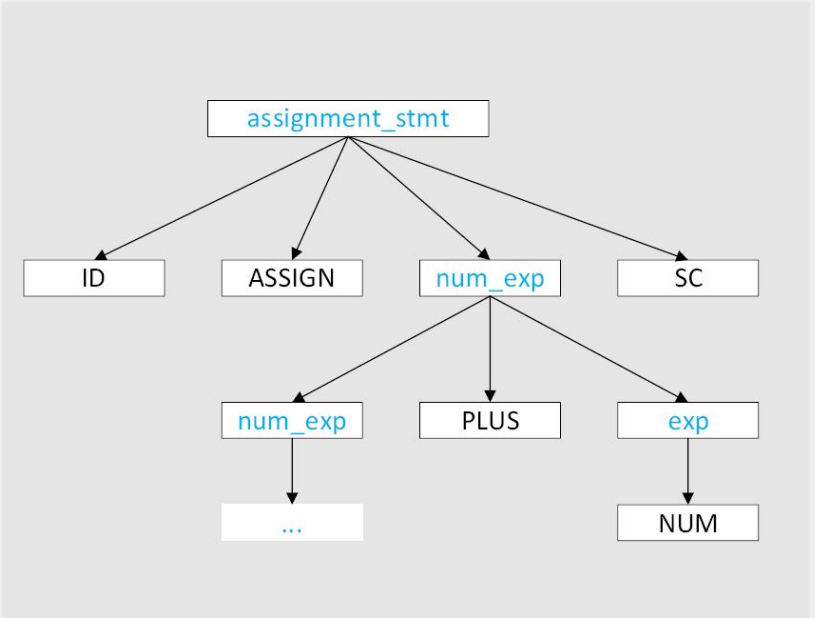
Programski prevodioci

04 Sintaksna analiza

Fakultet tehničkih nauka, Novi Sad
22-23/Z
Dunja Vrbaški

assignment_stmt → ID ASSIGN num_exp SC
num_exp → exp
num_exp → num_exp PLUS exp
num_exp → num_exp MINUS exp
exp → NUM
exp → ID

~~a = b + 3;~~
a = b + c + 3;
w = ID ASSIGN ID PLUS ID PLUS NUM SC



U prethodnom primeru prilično jasan odabir pravila.
Nekad, prilikom izvođenja, možemo krenuti različitim “putevima”.

Posmatrajmo sledeću gramatiku:

$E \rightarrow num$
$E \rightarrow E + E$

5 + 3 + 1
w = num plus num plus num

E → num
E → E + E

5 + 3 + 1
~~w = num plus num plus num~~
w = num + num + num

Radi bolje preglednosti u nastavku ćemo “plus” pisati kao “+”

E → num
E → E + E

5 + 3 + 1
w = num + num + num

E
⇒ E + E
⇒ num + E
⇒ num + E + E
⇒ num + num + E
⇒ num + num + num

E
⇒ E + E
⇒ E + num
⇒ E + E + num
⇒ E + num + num
⇒ num + num + num

Leftmost derivation (najlevlje izvođenje, izvođenje s leva)
Izvođenje u kom se uvek primenjuje pravilo na prvi pojam sa leve strane

Rightmost derivation (najdešnje izvođenje, izvođenje s desna)
Izvođenje u kom se uvek primenjuje pravilo na prvi pojam sa desne strane

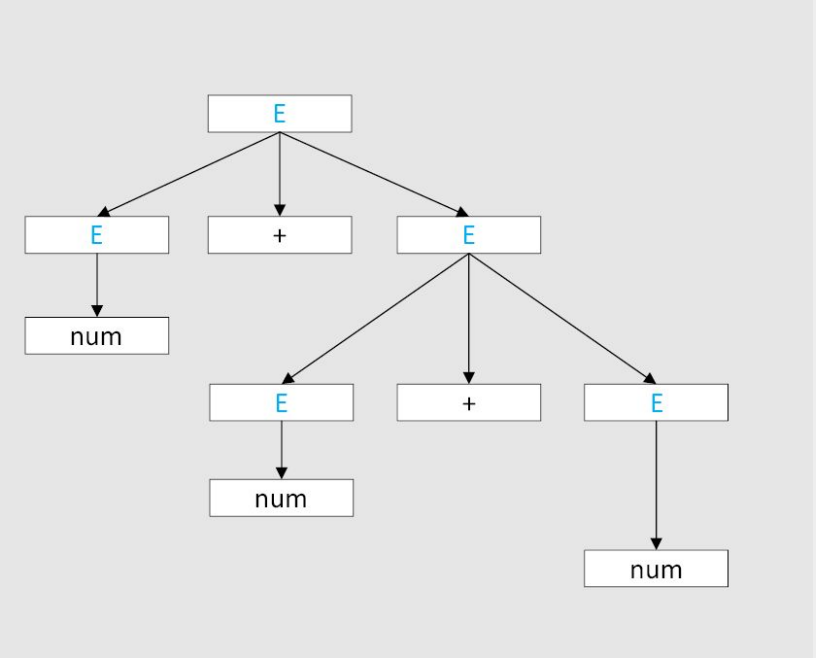
Postoje izvođenja koja ne prate ni jedno od ove dve heuristike.

Leftmost

E → num
E → E + E

5 + 3 + 1
w = num + num + num

E
⇒ E + E
⇒ num + E
⇒ num + E + E
⇒ num + num + E
⇒ num + num + num

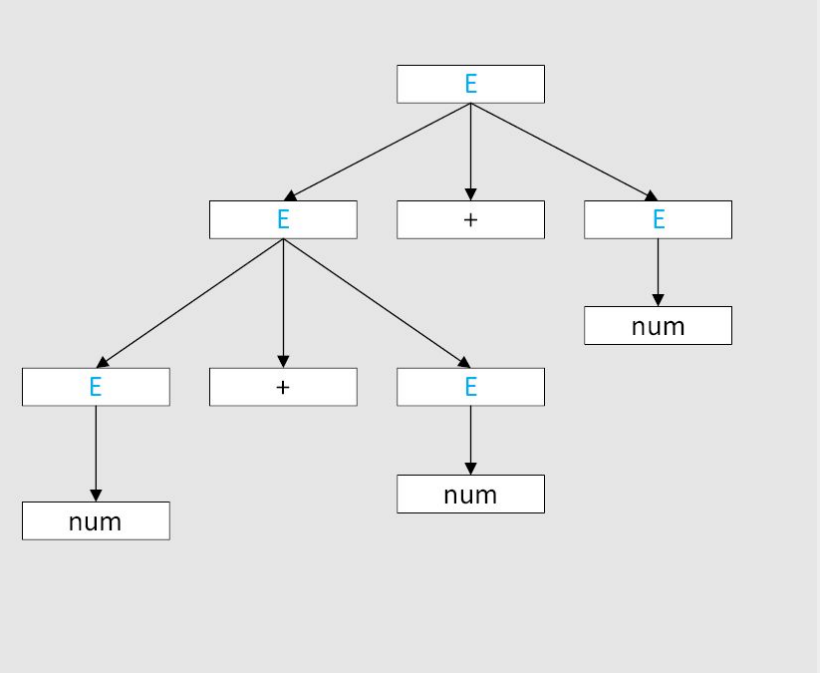


Rightmost

E → num
E → E + E

5 + 3 + 1
w = num + num + num

E
⇒ E + E
⇒ E + num
⇒ E + E + num
⇒ E + num + num
⇒ num + num + num

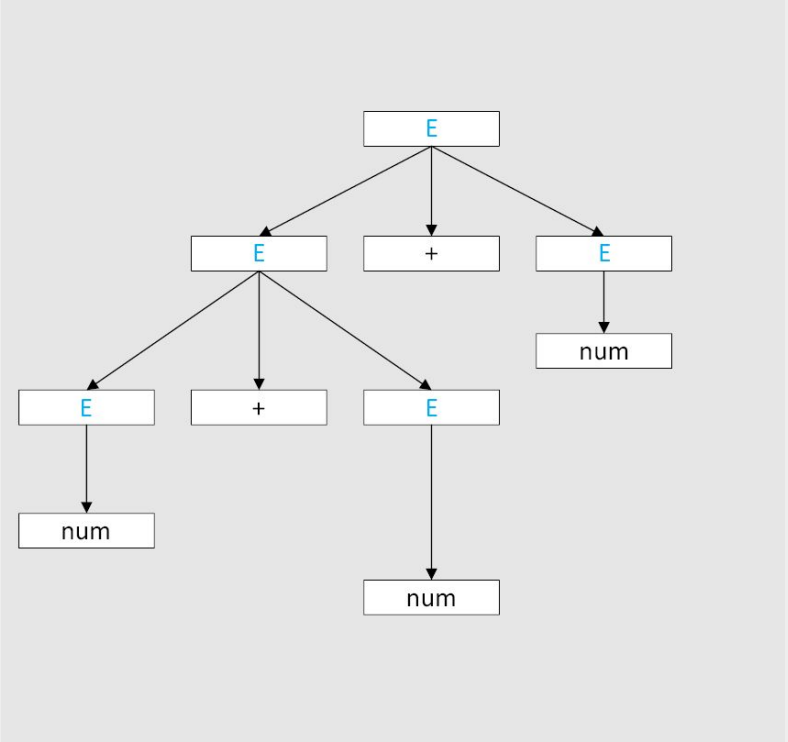


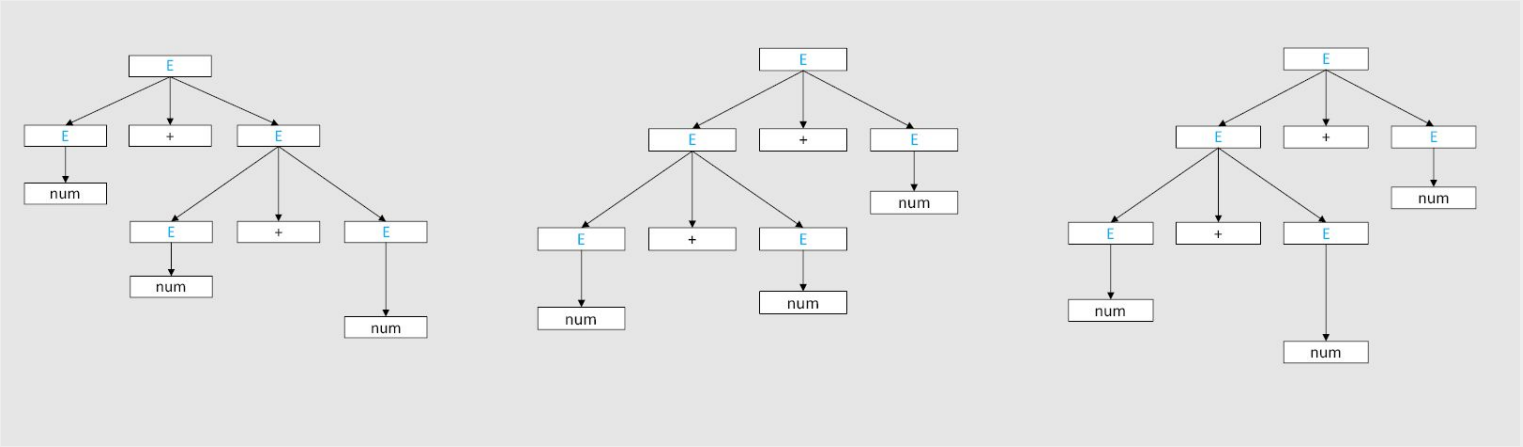
Ni L ni R

E → num
E → E + E

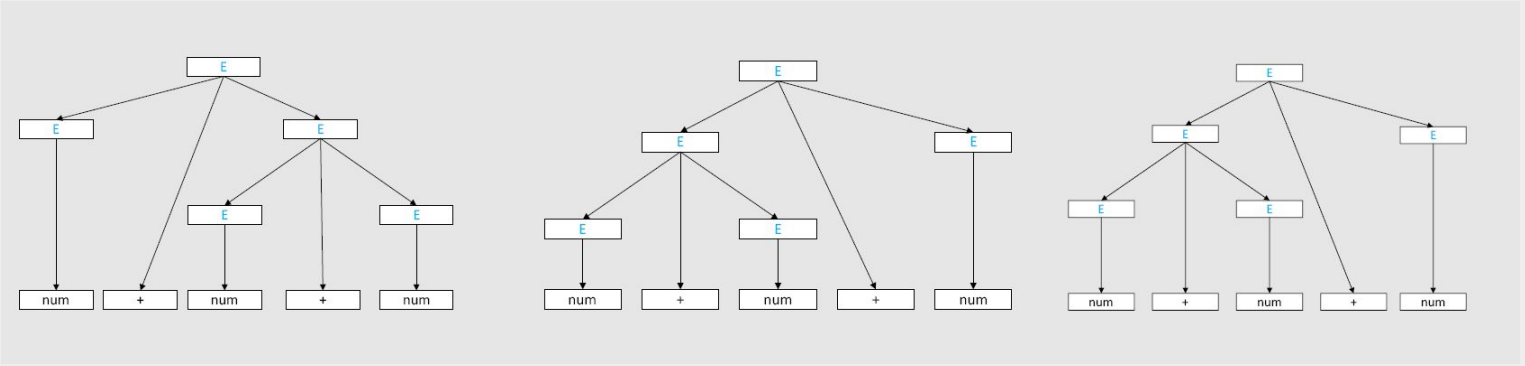
5 + 3 + 1
w = num + num + num

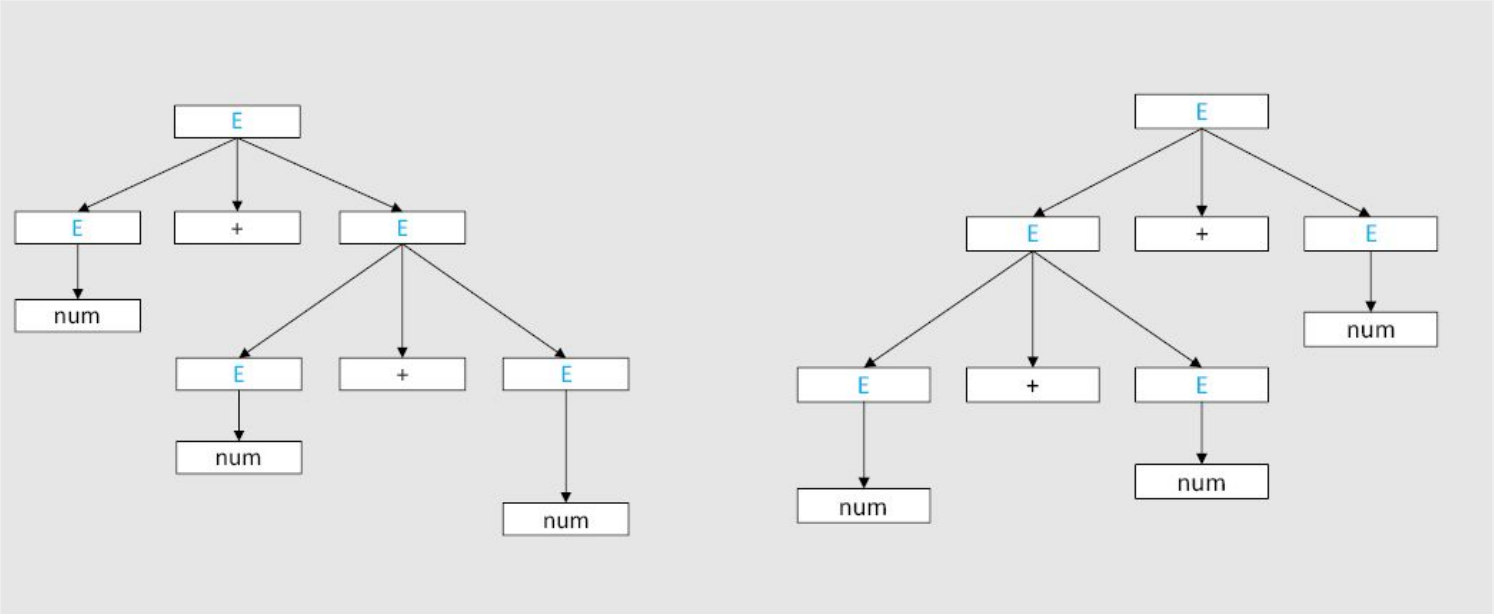
E
⇒ E + E
⇒ E + num
⇒ E + E + num
⇒ num + E + num
⇒ num + num + num





Drugo i treće stablo - zapravo ne postoji informacija o redosledu.
Na slikama je modelovan redosled (grafički spuštanjem listova), ali stablo, suštinski, nema tu informaciju (dužine strelica)


E
E
E
E
E
E
num
+
num
+
num
E
E
E
E
E
E
num
+
num
+
num
E
E
E
E
E
E
num
+
num
+
num



Može se primetiti da stablo modelira asocijativnost operacija
num + (num + num)
(num + num) + num

5 + 3 + 1

Ako je operator + levo asocijativan operand 3 će biti preuzet od strane levog + operatora
(5 + 3) + 1

Ako je operator + desno asocijativan operand 3 će bit preuzet od strane desnog + operatora
5 + (3 + 1)

Obratiti pažnju: ovde nije morao biti operator sabiranja niti su operandi nužno numeričke vrednosti.

Može biti značajno kod evaluacije.

Iako se trenutno ne bavimo evaluacijom, zanima nas samo struktura, razmisliti:
šta ako i kad bude trebala evaluacija ovog izraza (izračunavanje)?

Već sada to moramo imati rešeno.

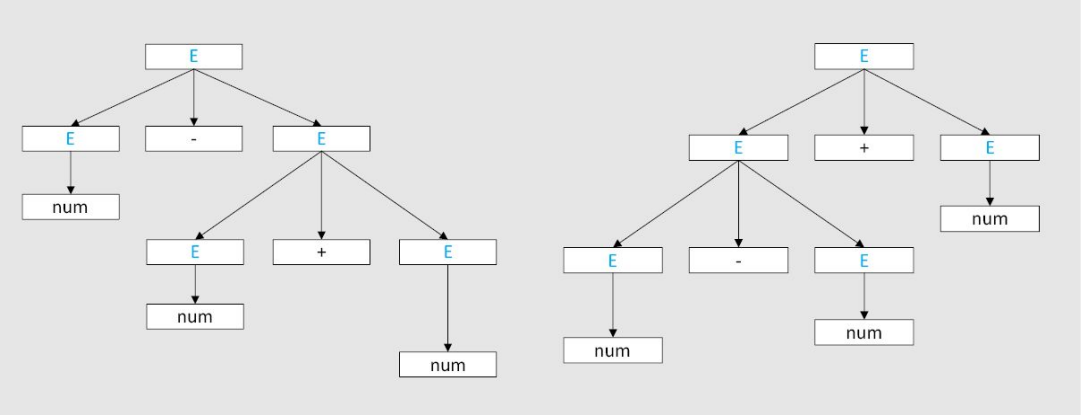
Posmatrajmo sledeću gramatiku

E → num
E → E + E
E → E - E

5 - 3 + 1
w = num - num + num

E → num
E → E + E
E → E - E

5 - 3 + 1
w = num - num + num



5 – (3 + 1)

(5 – 3) + 1

Obratiti pažnju: Gramatika nema zagrade. Ovde su iskorišćene samo kao prikaz potencijalnog redosleda evaluacije

Različita stabla parsiranja mogu odgovarati istom stringu.

Kažemo da je gramatika dvosmislena (višeznačna) ako za neki polazni string postoje bar dva različita stabla parsiranja.

Ako svaki string ima jedinstveno stablo parsiranja onda svaki string ima i jedinstveno najlevlje izvođenje i jedinstveno najdešnje izvođenje.

Ne postoji univerzalni algoritam za detekciju dvosmislene gramatike (odlučivanje) ili njenog pretvaranja u gramatiku koja nije dvosmislena.

Kako da rešimo?

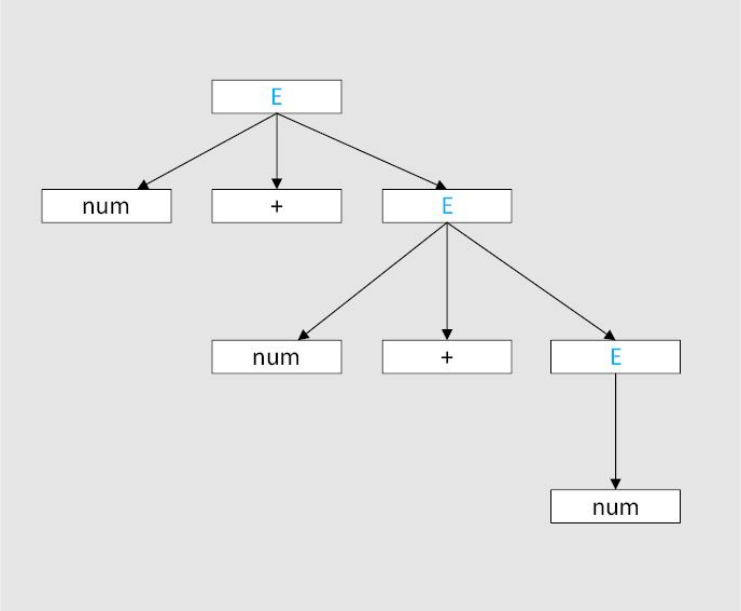
- Možemo promeniti gramatiku. Dvosmislenost je osobina gramatike – ne jezika!
- Koristeći neke mehanizme/ideje

~~E → num~~
~~E → E + E~~

E → num
E → num + E

5 + 3 + 1
w = num + num + num

E
⇒ num + E
⇒ num + num + E
⇒ num + num + num



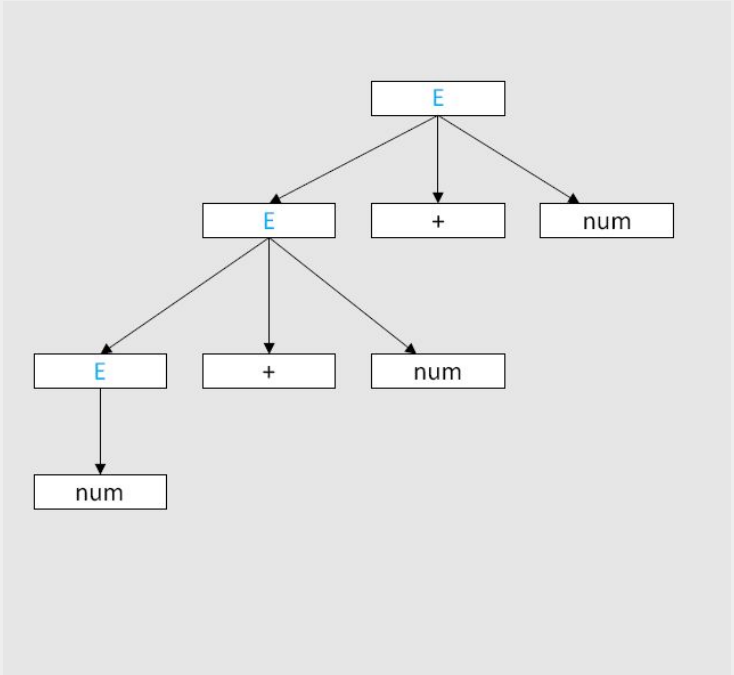
Obratiti pažnju: definisano kao lista

~~E → num~~
~~E → E + E~~

E → num
E → E + num

5 + 3 + 1
w = num + num + num

E
⇒ E + num
⇒ E + num + num
⇒ num + num + num



$E \rightarrow num$
$E \rightarrow E + E$

dvosmislena

$E \rightarrow num$
$E \rightarrow num + E$

desna rekurzija

$E \rightarrow num$
$E \rightarrow E + num$

leva rekurzija

Usput: dvosmislenost jeste problem, vrsta rekurzije može odrediti stablo, međutim vrsta rekurzije koja se pojavljuje može biti problem za određenu vrstu parsiranja.

Posmatrajmo sad sledeću gramatiku:

(na trenutak zaboravimo na to što imamo dvostruku rekurziju)

E → num
E → E + E
E → E * E

5 + 3 * 2
w = num + num * num

1. E → num
2. E → E + E
3. E → E * E

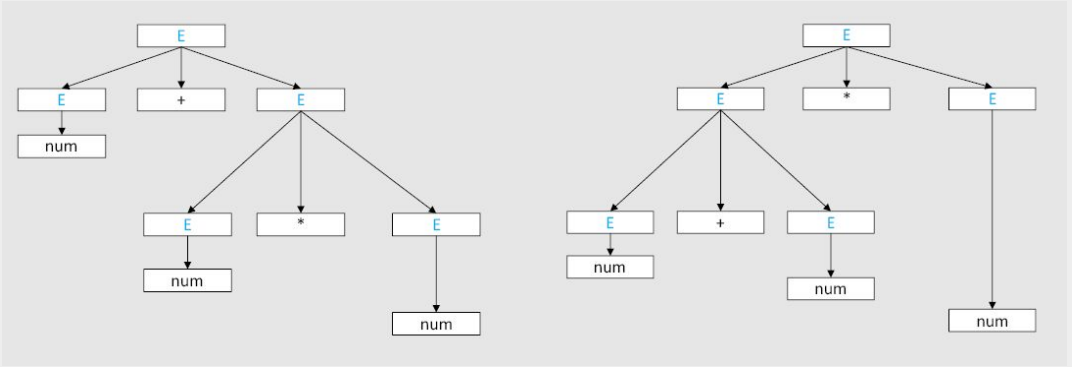
5 + 3 * 2
w = num + num * num

leftmost + prvo pravilo 2

E
⇒ E + E
⇒ num + E
⇒ num + E * E
⇒ num + num * E
⇒ num + num * num

leftmost + prvo pravilo 3

E
⇒ E * E
(ne može se menjati sa num jer nećemo stići do izraza koji tražimo)
⇒ E + E * E
⇒ num + E * E
⇒ num + num * E
⇒ num + num * num



5 + (3 *2)

(5 + 3) * 2

Ako smatramo da su osnovni aritmetički operatori levo asocijativni
→ to nam ne rešava problem.

Potreban je prioritet operatora.

Moramo obezbediti asocijativnost
Moramo obezbediti prioritet

Kako da rešimo?

- Možemo promeniti gramatiku
- Možemo možda nekako definisati prioritet (forsirati određena pravila).

Ideja:
5 + 3 * 2 + 4 * 1 …
Posmatramo kao sabirke

~~1. E → num~~
~~2. E → E + E~~
~~3. E → E * E~~

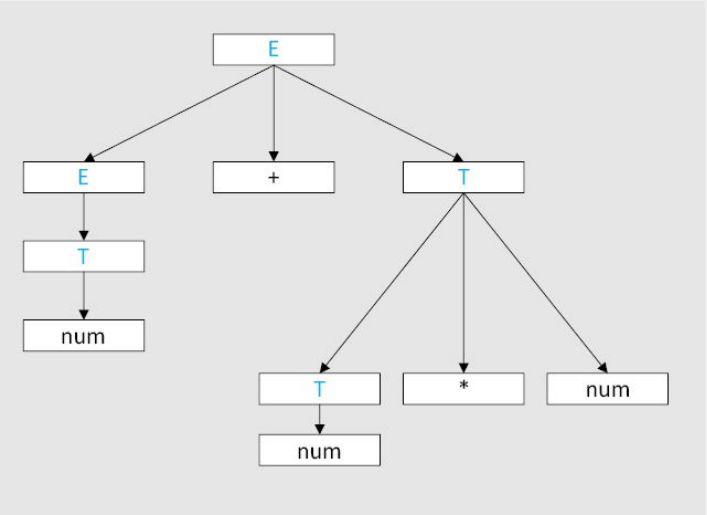
1. E → E + T
2. E → T
3. T → T * num
4. T → num

5 + 3 * 2
w = num + num * num

Obratiti pažnju: definisano kao liste

leftmost

E
⇒ E + T
⇒ T + T
⇒ num + T
⇒ num + T * num
⇒ num + num * num



miniC ima samo + i –
Ako se dodaje * i / - treba razmisliti o ovome

promena gramatike: dobra opcija

bison:
- operator naveden kasnije ima veći prioritet
- operatori navedeni u istom redu imaju isti prioritet
- postoji posebni mehanizmi za definisanje asocijativnosti i prioriteta (u nastavku)

Dangling else problem

stmt → assign_stmt
stmt → if_stmt
stmt → while_stmt
stmt → …

if_stmt → IF exp stmt
if_stmt → IF exp stmt ELSE stmt

ILI (kraće, liči na bison)

stmt	→	if_stmt
if_stmt	→	iF exp stmt
	|	IF exp stmt ELSE stmt

stmt → assign_stmt
stmt → if_stmt
stmt → …

if_stmt → IF exp stmt
if_stmt → IF exp stmt ELSE stmt

ILI (kraće, liči na bison)

stmt	→	if_stmt
if_stmt	→	iF exp stmt
	|	IF exp stmt ELSE stmt

```
if (a != b) if (a > b) b = a; else a = b;
```

Kako bismo želeli da se interpretira?

```
if (a != b)
   if (a > b)
      b = a;
   else
      a = b;
```

```
if (a != b)
   if (a > b)
      b = a;
else
    a = b;
```

stmt	→	if_stmt
if_stmt	→	iF exp stmt
	|	IF exp stmt ELSE stmt

```
if (a != b) if (a > b) b = a; else a = b;
```

w= IF LP ID ROP ID RP IF LP ID ROP ID ID ASGN ID SC ELSE ID ASGN ID SC

if_stmt
⇒ **IF exp stmt**
⇒ IF … stmt
⇒ IF … if_stmt
⇒ IF … IF exp stmt ELSE stmt
⇒ IF … IF … stmt ELSE stmt
⇒ IF … IF … … ELSE stmt
⇒ IF … IF … … ELSE …

if_stmt
⇒ **IF exp stmt ELSE stmt**
⇒ IF … stmt ELSE stmt
⇒ IF … if_stmt ELSE stmt
⇒ IF … IF exp stmt ELSE stmt
⇒ IF … IF … stmt ELSE stmt
⇒ IF … IF … … ELSE stmt
⇒ IF … IF … … ELSE …

Koje izvođenje je "tačno"? I zašto je ovo pod navodnicima?

shift-reduce parseri
(bison)

Ukoliko postoji višeznačnost javljaju se konflikti.
Potencijalno se, u datom stanju, može izvršiti više akcija.

shift-reduce konflikti
Kada, posmatrajući trenutni token, može da se izvrši i jedna i druga akcija

reduce-reduce konflikti
Kada postoji više pravila koja se mogu primeniti na ulazni niz.

Podrazumevano rešavanje

shift- reduce konflikti – prednost se daje shift akciji
reduce-reduce konflikti - prednost se daje reduce akciji čije pravilo je prvo navedeno

Podrazumevane akcije mogu biti neodgovarajuće.
Posebno obratiti pažnju na reduce-reduce konflikte.
Ukazuje na ozbiljan problem u gramatici.
Bolje proučiti i izmeniti gramatiku.

Postoje i mehanizmi koji se odnose na prioritet (precedence) i redosled primene (associativity) tokena/operatora.

Pitanja i zadaci

Implementirati bottom-up parser za kalkulator (flex/bison).

Implementirati ručno top-down parser za kalkulator.
U obe implementacije - inkrementalni razvoj:

- *Početi od +, dodati *, dodati – i /, dodati zagrade, dodati unarni minus.*
- *Revidirati gramatiku po potrebi.*
- *Pisati stalno test primere (ispravni i pogrešni ulazi). Ostavljati “sanity” test primere.*

Pokušati sa ispisom stabla parsiranja prilikom ovih implementacija (top-down, bottom-up)

Šta je inorder, preorder i postorder obilazak stabla?

Koji operator je desno asocijativan?