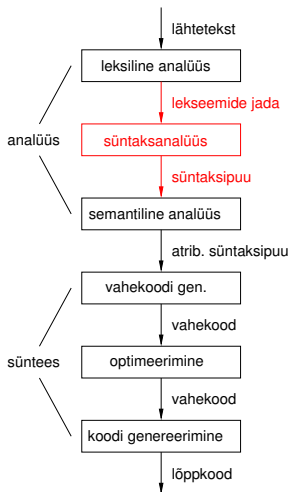


Süntaksanalüüs

Süntaksanalüüs

- **Süntaksanalüüs** kontrollib programmi struktuuri vastavust keele grammatikale:
 - saab sisendina, skanneri poolt genereeritud, lekseemide jada;
 - väljastab programmi esitava (abstraktse) süntaksipuu;
 - süntaktiliste vigade korral, teeb kindlaks nende asukoha;
 - ... teavitab võimalikest vea põhjustest;
 - ... püüab veast toibuda ja jätkata analüüsi (et järgnevaid vigu avastada).
- Süntaksanalüüsi kutsutakse **parsimiseks** (**parsing**) ning vastavat analüsaatorit nimetatakse **parseriks** (**parser**).



Grammatikad

- Keelte süntaksi kirjeldatakse reeglina kontekstivaba grammatika abil.
- **Grammatika** on nelik $G = \langle N, T, P, S \rangle$, kus
 - N on lõplik **mitteterminaalide** tähestik;
 - T on lõplik **terminaalsümbolite** tähestik;
 - $N \cap T = \emptyset$ ja $V = N \cup T$;
 - $P \subset \{\alpha \rightarrow \beta \mid \alpha \in V^+, \beta \in V^*\}$ on lõplik **produksioonireeglite** hulk;
 - $S \in N$ on **algsümbol**.
- Grammatika on **kontekstivaba** (**context-free**), kui produktsioonireeglid on kujul $A \rightarrow \alpha$, kus $A \in N$ ja $\alpha \in V^*$.

Grammatikad

- Jada $w \in V^*$ nimetatakse **lausevormiks** (sentential form).
- Lausevorm $v \in V^*$ on **otsetuletatav** (directly derivable) lausevormist $u \in V^*$ (tähistus $u \Rightarrow v$), kui leiduvad $w_1, w_2, \alpha, \beta \in V^*$ sellised, et $u = w_1\alpha w_2$, $v = w_1\beta w_2$ ja $\alpha \rightarrow \beta \in P$.
- Relatsiooni \Rightarrow refleksiivset transitiivset sulundit (tähistus \Rightarrow^*) nimetatakse **derivatsiooniks** (derivation) ehk **tuletuseks**.
- Grammatika $G = \langle N, T, P, S \rangle$ genereerib **keele**

$$L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$$

- Grammatikad G_1 ja G_2 on **ekvivalentsed**, kui $L(G_1) = L(G_2)$.

Grammatikad

Chomsky hierarhia:

	Produksioonid	Keelte tüüp	Automaat
L_0	$\alpha \rightarrow \beta$	Semi-Thue süsteemid	Turingi masin
L_1	$\alpha A \beta \rightarrow \alpha \gamma \beta$	Kontekstist sõltuvad	Tõkestatud Turingi masin
L_2	$A \rightarrow \alpha$	Kontekstivabad	Magasinmäluga automaat
L_3	$A \rightarrow w, A \rightarrow wB$	Regulaarsed	Lõplik automaat
(L_4)	$A \rightarrow w$	Lõplikud	Tsükliteta lõplik automaat

kus $A, B \in N$, $\alpha, \beta, \gamma \in V^*$ ja $w \in T^*$.

Lemma: Chomsky hierarhia on range; so.:

$$(L_4) \subset L_3 \subset L_2 \subset L_1 \subset L_0$$

Kontekstivabad grammatikad

- Edaspidi käsitleme ainult kontekstivabu grammatikaid.
- Kontekstivabade grammatikate produktsioonireegleid esitatakse tavaliselt **Backus-Naur'i kujul** (BNF).
- Näide: olgu $N = \{\text{Exp}\}$ ja $T = \{+, *, (,), id\}$, siis

$$\begin{array}{l} \text{Exp} \rightarrow \text{Exp} + \text{Exp} \\ | \text{Exp} * \text{Exp} \\ | (\text{Exp}) \\ | id \end{array}$$

esitab produktsioonireeglite hulka

$$P = \{ \text{Exp} \rightarrow \text{Exp} + \text{Exp}, \text{Exp} \rightarrow (\text{Exp}), \\ \text{Exp} \rightarrow \text{Exp} * \text{Exp}, \text{Exp} \rightarrow id \}.$$

Kontekstivabad grammatikad

- Reeglina saab ühte ja sama lauset tuletada paljudel eri viisidel.
- Kanoonilised derivatsioonid:
 - **vasakderivatsioon** – derivatsiooni igal sammul asendatakse vasakpoolseim mitteterminaal;
 - **paremderivatsioon** – derivatsiooni igal sammul asendatakse parempoolseim mitteterminaal.
- Näide:

$$\begin{aligned} \text{Exp} &\Longrightarrow_{lm} \text{Exp} + \text{Exp} \\ &\Longrightarrow_{lm} id + \text{Exp} \\ &\Longrightarrow_{lm} id + \text{Exp} * \text{Exp} \\ &\Longrightarrow_{lm} id + id * \text{Exp} \\ &\Longrightarrow_{lm} id + id * id \end{aligned}$$

$$\begin{aligned} \text{Exp} &\Longrightarrow_{rm} \text{Exp} + \text{Exp} \\ &\Longrightarrow_{rm} \text{Exp} + \text{Exp} * \text{Exp} \\ &\Longrightarrow_{rm} \text{Exp} + \text{Exp} * id \\ &\Longrightarrow_{rm} \text{Exp} + id * id \\ &\Longrightarrow_{rm} id + id * id \end{aligned}$$

Kontekstivabad grammatikad

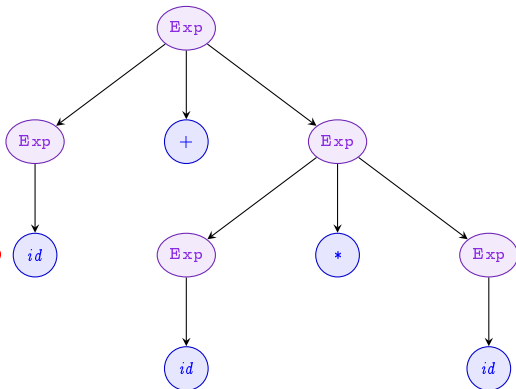
- Iga derivatsioon määrab üheselt **süntaksipuu** (**syntax-tree**, **parse-tree**), mis on järjestatud tippudega puu, kus:
 - puu juur on märgendatud algsümboliga S ;
 - vahetipud on märgendatud mitteterminaalidega;
 - lehed on märgendatud terminaalsümboliga või tühisümboliga ϵ ;
 - kui vahetipp on märgendatud mitteterminaaliga A ja tema alampuude (vasakult paremale) t_1, \dots, t_n juured on märgendatud vastavalt A_1, \dots, A_n , siis $A \rightarrow A_1 \dots A_n \in P$.
- Tuletatud lause saadakse lugedes puu lehtede märgendid vasakult paremale.
- Süntaksipuu määrab üheselt kasutatud produktsiooni-reeglid, kuid mitte nende rakendamise järjekorda.

Kontekstivabad grammatikad

Näide: eelnevalt toodud vasak- ja paremderivatsioonile vastab mõlemal juhul sama süntaksipuu

Exp \Rightarrow_{lm} **Exp** + **Exp**
 \Rightarrow_{lm} *id* + **Exp**
 \Rightarrow_{lm} *id* + **Exp** * **Exp**
 \Rightarrow_{lm} *id* + *id* * **Exp**
 \Rightarrow_{lm} *id* + *id* * *id*

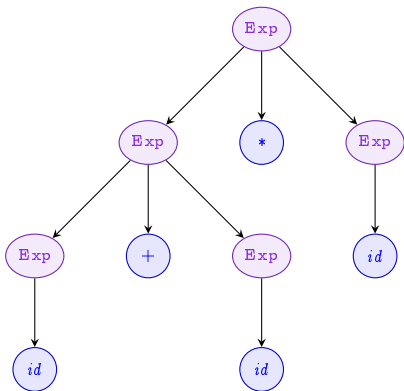
Exp \Rightarrow_{rm} **Exp** + **Exp**
 \Rightarrow_{rm} **Exp** + **Exp** * **Exp**
 \Rightarrow_{rm} **Exp** + **Exp** * *id*
 \Rightarrow_{rm} **Exp** + *id* * *id*
 \Rightarrow_{rm} *id* + *id* * *id*



Kontekstivabad grammatikad

NB! Ühel lausel võib olla mitu erinevat süntaksipuud!

$\text{Exp} \Rightarrow_{lm} \text{Exp} * \text{Exp}$
 $\Rightarrow_{lm} \text{Exp} + \text{Exp} * \text{Exp}$
 $\Rightarrow_{lm} id + \text{Exp} * \text{Exp}$
 $\Rightarrow_{lm} id + id * \text{Exp}$
 $\Rightarrow_{lm} id + id * id$



Kontekstivabad grammatikad

- KV-grammatika on **mitmene** (**ambiguous**), kui ühe lause jaoks leidub mitu süntaksipuud.
- Iga süntaksipuu jaoks leidub täpselt üks vasak- ja üks parenderivatsioon; seega:
 - ühesel lausel on täpselt üks vasak- ja üks parenderivatsioon;
 - mitmesel lausel on vähemalt kaks vasak- ja parenderivatsiooni.
- Lause erinevad süntaksipuud vastavad reeglina lause semantiliselt erinevatele tõlgendusvõimalustele.
- Mitmese grammatika saab teatud juhtudel (aga mitte alati) teisendada ekvivalentseks üheseks grammatikaks.

Kontekstivabad grammatikad

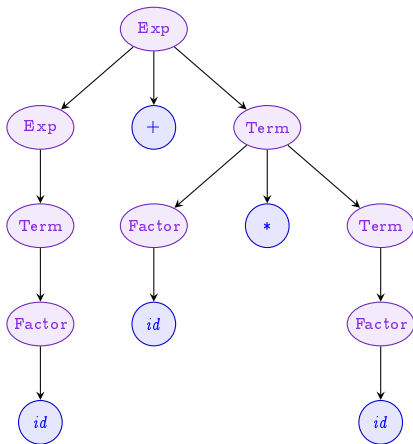
- Mitmesuse eemaldamine – binaarsed operaatorid:
 - iga prioriteeditaseme jaoks toome sisse uue mitteterminaali;
 - vasakassotsiatiivse operaatori korral kasutame vasakrekursiivset; paremassotsiatiivse korral paremrekursiivset produktsioonireeglit;
 - tugevama prioriteediga operaatoritele vastavad reeglid paigutame "sügavamale".
- Näide:

$$\begin{array}{l} \text{Exp} \quad \rightarrow \quad \text{Exp} + \text{Term} \\ \quad \quad \quad | \quad \text{Term} \\ \text{Term} \quad \rightarrow \quad \text{Factor} * \text{Term} \\ \quad \quad \quad | \quad \text{Factor} \\ \text{Factor} \quad \rightarrow \quad (\text{Exp}) \\ \quad \quad \quad | \quad \textit{id} \end{array}$$

Kontekstivabad grammatikad

Näide:

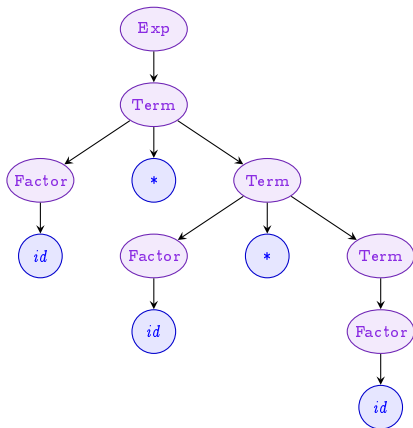
$\text{Exp} \Rightarrow_{lm} \text{Exp} + \text{Term}$
 $\Rightarrow_{lm} \text{Term} + \text{Term}$
 $\Rightarrow_{lm} \text{Factor} + \text{Term}$
 $\Rightarrow_{lm} id + \text{Term}$
 $\Rightarrow_{lm} id + \text{Factor} * \text{Term}$
 $\Rightarrow_{lm} id + id * \text{Term}$
 $\Rightarrow_{lm} id + id * \text{Factor}$
 $\Rightarrow_{lm} id + id * id$



Kontekstivabad grammatikad

Näide:

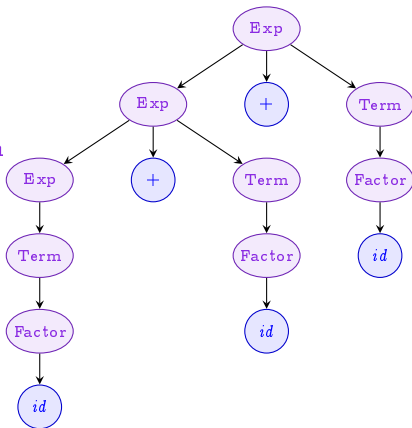
$\text{Exp} \Rightarrow_{lm} \text{Term}$
 $\Rightarrow_{lm} \text{Factor} * \text{Term}$
 $\Rightarrow_{lm} id * \text{Term}$
 $\Rightarrow_{lm} id * \text{Factor} * \text{Term}$
 $\Rightarrow_{lm} id * id * \text{Term}$
 $\Rightarrow_{lm} id * id * \text{Factor}$
 $\Rightarrow_{lm} id * id * id$



Kontekstivabad grammatikad

Näide:

$\text{Exp} \Rightarrow_{lm} \text{Exp} + \text{Term}$
 $\Rightarrow_{lm} \text{Exp} + \text{Term} + \text{Term}$
 $\Rightarrow_{lm} \text{Term} + \text{Term} + \text{Term}$
 $\Rightarrow_{lm} \text{Factor} + \text{Term} + \text{Term}$
 $\Rightarrow_{lm} id + \text{Term} + \text{Term}$
 $\Rightarrow_{lm} id + \text{Factor} + \text{Term}$
 $\Rightarrow_{lm} id + id + \text{Term}$
 $\Rightarrow_{lm} id + id + \text{Factor}$
 $\Rightarrow_{lm} id + id + id$



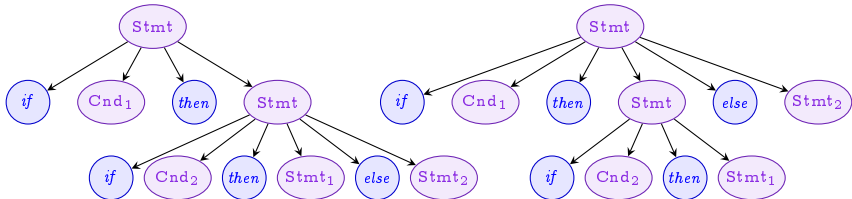
Kontekstivabad grammatikad

- Mitmesuse eemaldamine – tingimuslaused:

$\text{Stmt} \rightarrow \text{if Cnd then Stmt}$
 $\quad \quad \quad | \text{if Cnd then Stmt else Stmt}$
 $\quad \quad \quad | \text{Other}$

- Järgnev lausevorm omab kahte süntaksipuud:

$\text{if Cnd}_1 \text{ then if Cnd}_2 \text{ then Stmt}_1 \text{ else Stmt}_2$

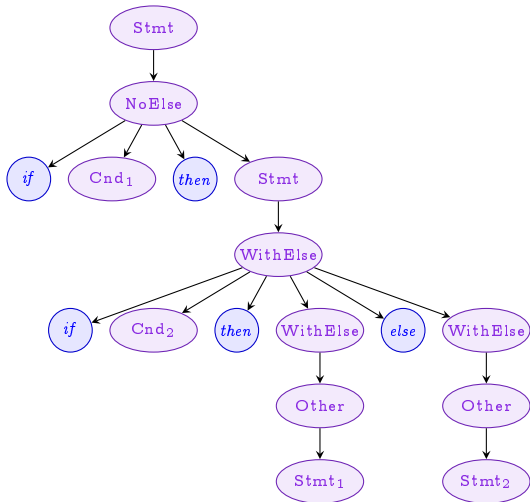


Kontekstivabad grammatikad

Tavaliselt loetakse korrektseks esimest süntaksipuud; so. *else* kuulub alati kõige sisemise võimaliku tingimuslause juurde:

```
Stmt      →  WithElse
           |  NoElse
WithElse  →  if Cnd then WithElse else WithElse
           |  Other
NoElse    →  if Cnd then Stmt
           |  if Cnd then WithElse else NoElse
```

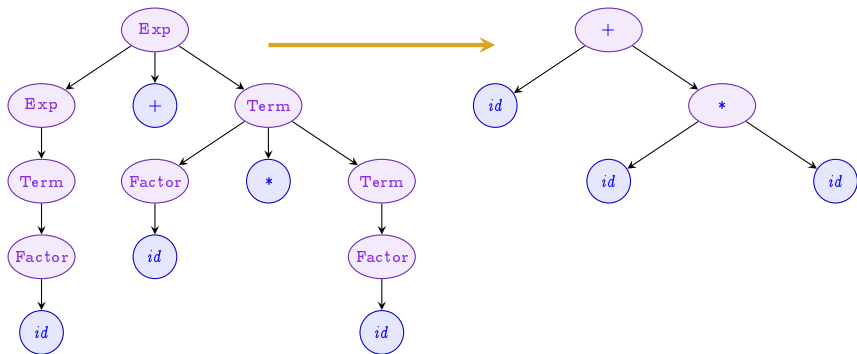
Kontekstivabad grammatikad



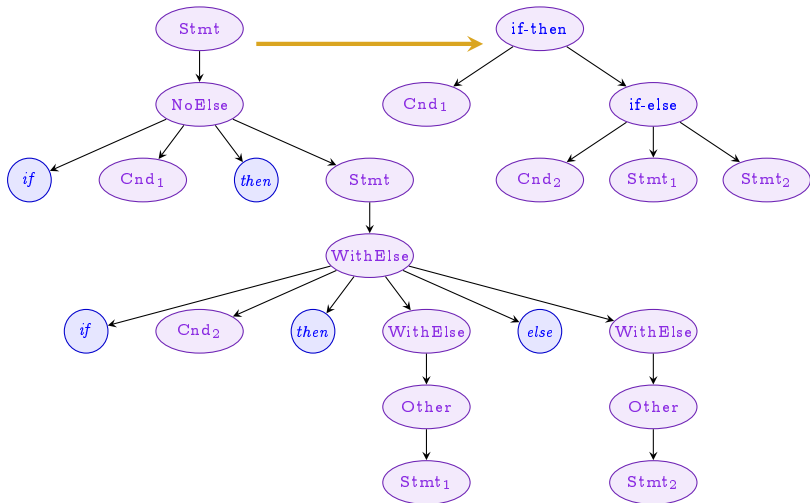
Abstrakne süntaksipuu

- Konkreetsele derivatsioonile vastava süntaksipuu sisaldab reeglina edasiseks transleerimiseks ebaolulist infot.
- Seetõttu tavaliselt konstrueeritakse parseris kompaktsem **abstrakne süntaksipuu** (**abstract syntax tree**, **AST**):
 - ebaolulised vahetipud on eemaldatud;
 - eri liiki tipud vastavad erinevatele keelekonstruktsioonidele.

Abstrakne süntaksipuu



Abstrakne süntaksipuu



Parsimistehnikad

Parsimistehnikad

Ülalt-alla parsimine (top-down parsing):

- alustab süntaksipuu ehitamist juurest ning kasvatab seda lehtede suunas;
- igal sammul valib produktsioonireegli ning üritab seda sobitada sisendsõnaga;
- mittesobiva reegli korral toimub tagasipöördumine (**backtracking**);
- reeglina annab vasakpoolse derivatsiooni.

Alt-üles parsimine (bottom-up parsing)

- alustab süntaksipuu ehitamist lehtedest ning kasvatab seda juure suunas;
- rakendab sobivaid reegleid paremalt vasakule kuni jõuab algsümbolini;
- reeglina annab parempoolse derivatsiooni.

Ülalt-alla parsimine

Ülalt-alla parsimise üldine algoritm:

- konstrueerime juurtipu, märgendame selle algsümboliga ja jätkame puu konstrueerimist lehteded suunas sügavuti vasakult paremale;
- kui vaatluse all olev tipp on mitteterminaal A , siis valime mingi produktsioonireegli kujul $A \rightarrow \alpha$, konstrueerime reegli paremale poolele vastavad tipud, ning jätkame vasakpoolseima alamtipuga;
- kui tipp on terminaalsümbol mis ei sobi sisendsümboliga, siis teostame tagasipöördumise produktsioonireegli valikuni mis selle tipu konstrueeris ja jätkame sealt uuesti valides teise reegli;
- kui tipp on terminaalsümbol mis sobib sisendsümboliga, siis jätkame järgmise vasakpoolseima läbivaatamata tipuga.

Ülalt-alla parsimine

Näide:

E	\rightarrow	$E + T$	\bullet	$id - num * id$	
		$E - T$	$E.1$	\bullet	$id - num * id$
		T	$E.3$	\bullet	$id - num * id$
T	\rightarrow	$T * F$	$T.3$	\bullet	$id - num * id$
		T / F	$F.2$	\bullet	$id - num * id$
		F		$id \bullet - num * id$	
F	\rightarrow	(E)		$id - \bullet num * id$	
		id	$T.1$	$id - \bullet num * id$	
		num	$T.3$	$id - \bullet num * id$	
			$F.3$	$id - \bullet num * id$	
				$id - num \bullet * id$	
				$id - num * \bullet id$	
			$F.2$	$id - num * \bullet id$	
				$id - num * id \bullet$	

Ülalt-alla parsimine

Näide:

$E \rightarrow E + T$
| $E - T$
| T
 $T \rightarrow T * F$
| T / F
| F
 $F \rightarrow (E)$
| id
| num

• $id - num * id$
 $E.1$ • $id - num * id$
 $E.3$ • $id - num * id$
 $T.3$ • $id - num * id$
 $F.2$ • $id - num * id$
 $id \bullet - num * id$
 $id - \bullet num * id$
 $T.1$ $id - \bullet num * id$
 $T.3$ $id - \bullet num * id$
 $F.3$ $id - \bullet num * id$
 $id - num \bullet * id$
 $id - num * \bullet id$
 $F.2$ $id - num * \bullet id$
 $id - num * id \bullet$

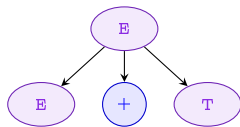
E

Ülalt-alla parsimine

Näide:

$E \rightarrow E + T$
 $\quad | E - T$
 $\quad | T$
 $T \rightarrow T * F$
 $\quad | T / F$
 $\quad | F$
 $F \rightarrow (E)$
 $\quad | id$
 $\quad | num$

$\bullet id - num * id$
E.1 $\bullet id - num * id$
E.3 $\bullet id - num * id$
T.3 $\bullet id - num * id$
F.2 $\bullet id - num * id$
 $id \bullet - num * id$
 $id - \bullet num * id$
T.1 $id - \bullet num * id$
T.3 $id - \bullet num * id$
F.3 $id - \bullet num * id$
 $id - num \bullet * id$
 $id - num * \bullet id$
F.2 $id - num * \bullet id$
 $id - num * id \bullet$

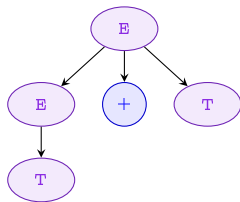


Ülalt-alla parsimine

Näide:

$E \rightarrow E + T$
| $E - T$
| T
 $T \rightarrow T * F$
| T / F
| F
 $F \rightarrow (E)$
| id
| num

• $id - num * id$
 $E.1$ • $id - num * id$
 $E.3$ • $id - num * id$
 $T.3$ • $id - num * id$
 $F.2$ • $id - num * id$
 $id \bullet - num * id$
 $id - \bullet num * id$
 $T.1$ $id - \bullet num * id$
 $T.3$ $id - \bullet num * id$
 $F.3$ $id - \bullet num * id$
 $id - num \bullet * id$
 $id - num * \bullet id$
 $F.2$ $id - num * \bullet id$
 $id - num * id \bullet$

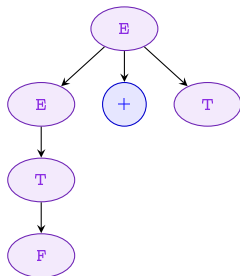


Ülalt-alla parsimine

Näide:

$E \rightarrow E + T$
 $\quad | \quad E - T$
 $\quad | \quad T$
 $T \rightarrow T * F$
 $\quad | \quad T / F$
 $\quad | \quad F$
 $F \rightarrow (E)$
 $\quad | \quad id$
 $\quad | \quad num$

$\bullet id - num * id$
E.1 $\bullet id - num * id$
E.3 $\bullet id - num * id$
T.3 $\bullet id - num * id$
F.2 $\bullet id - num * id$
 $id \bullet - num * id$
 $id - \bullet num * id$
T.1 $id - \bullet num * id$
T.3 $id - \bullet num * id$
F.3 $id - \bullet num * id$
 $id - num \bullet * id$
 $id - num * \bullet id$
F.2 $id - num * \bullet id$
 $id - num * id \bullet$

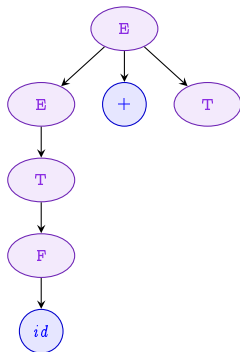


Ülalt-alla parsimine

Näide:

$E \rightarrow E + T$
| $E - T$
| T
 $T \rightarrow T * F$
| T / F
| F
 $F \rightarrow (E)$
| id
| num

• $id - num * id$
 $E.1$ • $id - num * id$
 $E.3$ • $id - num * id$
 $T.3$ • $id - num * id$
 $F.2$ • $id - num * id$
 $id \bullet - num * id$
 $id - \bullet num * id$
 $T.1$ $id - \bullet num * id$
 $T.3$ $id - \bullet num * id$
 $F.3$ $id - \bullet num * id$
 $id - num \bullet * id$
 $id - num * \bullet id$
 $F.2$ $id - num * \bullet id$
 $id - num * id \bullet$

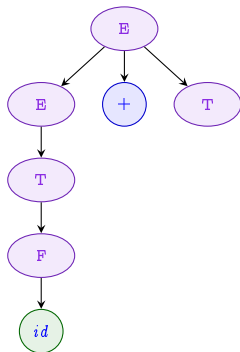


Ülalt-alla parsimine

Näide:

$E \rightarrow E + T$
 $E \rightarrow E - T$
 $E \rightarrow T$
 $T \rightarrow T * F$
 $T \rightarrow T / F$
 $T \rightarrow F$
 $F \rightarrow (E)$
 $F \rightarrow id$
 $F \rightarrow num$

$\bullet id - num * id$
E.1 $\bullet id - num * id$
E.3 $\bullet id - num * id$
T.3 $\bullet id - num * id$
F.2 $\bullet id - num * id$
 $id \bullet - num * id$
 $id - \bullet num * id$
T.1 $id - \bullet num * id$
T.3 $id - \bullet num * id$
F.3 $id - \bullet num * id$
 $id - num \bullet * id$
 $id - num * \bullet id$
F.2 $id - num * \bullet id$
 $id - num * id \bullet$

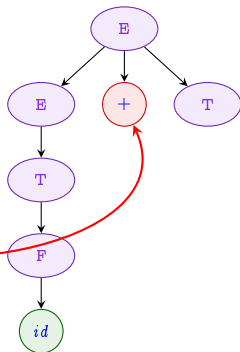


Ülalt-alla parsimine

Näide:

$E \rightarrow E + T$
 $\quad | \quad E - T$
 $\quad | \quad T$
 $T \rightarrow T * F$
 $\quad | \quad T / F$
 $\quad | \quad F$
 $F \rightarrow (E)$
 $\quad | \quad id$
 $\quad | \quad num$

$\bullet id - num * id$
 $E.1 \bullet id - num * id$
 $E.3 \bullet id - num * id$
 $T.3 \bullet id - num * id$
 $F.2 \bullet id - num * id$
 $id \bullet (-) num * id$
 $id - \bullet num * id$
 $T.1 id - \bullet num * id$
 $T.3 id - \bullet num * id$
 $F.3 id - \bullet num * id$
 $id - num \bullet * id$
 $id - num * \bullet id$
 $F.2 id - num * \bullet id$
 $id - num * id \bullet$



Ülalt-alla parsimine

Näide:

$E \rightarrow E + T$
| $E - T$
| T
 $T \rightarrow T * F$
| T / F
| F
 $F \rightarrow (E)$
| id
| num

• $id - num * id$
 $E.1$ • $id - num * id$
 $E.3$ • $id - num * id$
 $T.3$ • $id - num * id$
 $F.2$ • $id - num * id$
 $id \bullet - num * id$
 $id - \bullet num * id$
 $T.1$ $id - \bullet num * id$
 $T.3$ $id - \bullet num * id$
 $F.3$ $id - \bullet num * id$
 $id - num \bullet * id$
 $id - num * \bullet id$
 $F.2$ $id - num * \bullet id$
 $id - num * id \bullet$

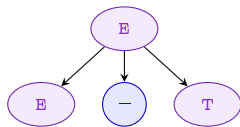
E

Ülalt-alla parsimine

Näide:

$E \rightarrow E + T$
| $E - T$
| T
 $T \rightarrow T * F$
| T / F
| F
 $F \rightarrow (E)$
| id
| num

• $id - num * id$
 $E.2$ • $id - num * id$
 $E.3$ • $id - num * id$
 $T.3$ • $id - num * id$
 $F.2$ • $id - num * id$
 $id \bullet - num * id$
 $id - \bullet num * id$
 $T.1$ $id - \bullet num * id$
 $T.3$ $id - \bullet num * id$
 $F.3$ $id - \bullet num * id$
 $id - num \bullet * id$
 $id - num * \bullet id$
 $F.2$ $id - num * \bullet id$
 $id - num * id \bullet$

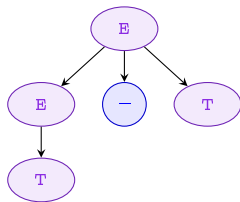


Ülalt-alla parsimine

Näide:

$E \rightarrow E + T$
 $\quad | \quad E - T$
 $\quad | \quad T$
 $T \rightarrow T * F$
 $\quad | \quad T / F$
 $\quad | \quad F$
 $F \rightarrow (E)$
 $\quad | \quad id$
 $\quad | \quad num$

$\bullet id - num * id$
 $E.2 \bullet id - num * id$
 $E.3 \bullet id - num * id$
 $T.3 \bullet id - num * id$
 $F.2 \bullet id - num * id$
 $id \bullet - num * id$
 $id - \bullet num * id$
 $T.1 id - \bullet num * id$
 $T.3 id - \bullet num * id$
 $F.3 id - \bullet num * id$
 $id - num \bullet * id$
 $id - num * \bullet id$
 $F.2 id - num * \bullet id$
 $id - num * id \bullet$

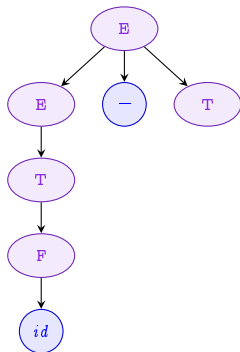


Ülalt-alla parsimine

Näide:

$E \rightarrow E + T$
| $E - T$
| T
 $T \rightarrow T * F$
| T / F
| F
 $F \rightarrow (E)$
| id
| num

• $id - num * id$
 $E.2$ • $id - num * id$
 $E.3$ • $id - num * id$
 $T.3$ • $id - num * id$
 $F.2$ • $id - num * id$
 $id \bullet - num * id$
 $id - \bullet num * id$
 $T.1$ $id - \bullet num * id$
 $T.3$ $id - \bullet num * id$
 $F.3$ $id - \bullet num * id$
 $id - num \bullet * id$
 $id - num * \bullet id$
 $F.2$ $id - num * \bullet id$
 $id - num * id \bullet$

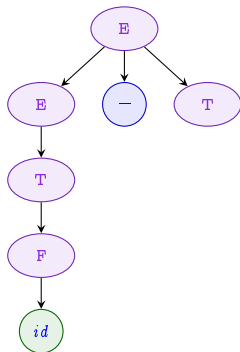


Ülalt-alla parsimine

Näide:

$E \rightarrow E + T$
 $\quad | E - T$
 $\quad | T$
 $T \rightarrow T * F$
 $\quad | T / F$
 $\quad | F$
 $F \rightarrow (E)$
 $\quad | id$
 $\quad | num$

$\bullet id - num * id$
 $E.2 \bullet id - num * id$
 $E.3 \bullet id - num * id$
 $T.3 \bullet id - num * id$
 $F.2 \bullet id - num * id$
 $id \bullet - num * id$
 $id - \bullet num * id$
 $T.1 id - \bullet num * id$
 $T.3 id - \bullet num * id$
 $F.3 id - \bullet num * id$
 $id - num \bullet * id$
 $id - num * \bullet id$
 $F.2 id - num * \bullet id$
 $id - num * id \bullet$

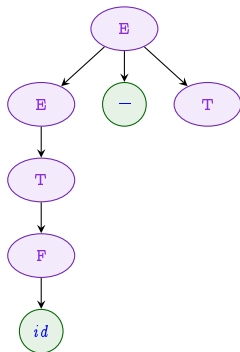


Ülalt-alla parsimine

Näide:

$E \rightarrow E + T$
 $E \rightarrow E - T$
 $E \rightarrow T$
 $T \rightarrow T * F$
 $T \rightarrow T / F$
 $T \rightarrow F$
 $F \rightarrow (E)$
 $F \rightarrow id$
 $F \rightarrow num$

$\bullet id - num * id$
E.2 $\bullet id - num * id$
E.3 $\bullet id - num * id$
T.3 $\bullet id - num * id$
F.2 $\bullet id - num * id$
 $id \bullet - num * id$
 $id - \bullet num * id$
T.1 $id - \bullet num * id$
T.3 $id - \bullet num * id$
F.3 $id - \bullet num * id$
 $id - num \bullet * id$
 $id - num * \bullet id$
F.2 $id - num * \bullet id$
 $id - num * id \bullet$

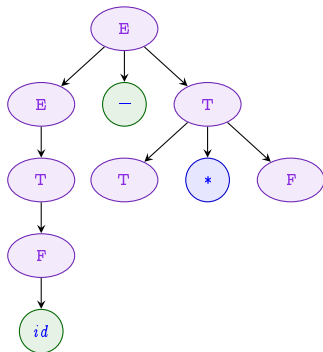


Ülalt-alla parsimine

Näide:

$E \rightarrow E + T$
 $\quad | E - T$
 $\quad | T$
 $T \rightarrow T * F$
 $\quad | T / F$
 $\quad | F$
 $F \rightarrow (E)$
 $\quad | id$
 $\quad | num$

$\bullet id - num * id$
 $E.2 \bullet id - num * id$
 $E.3 \bullet id - num * id$
 $T.3 \bullet id - num * id$
 $F.2 \bullet id - num * id$
 $id \bullet - num * id$
 $id - \bullet num * id$
 $T.1 id - \bullet num * id$
 $T.3 id - \bullet num * id$
 $F.3 id - \bullet num * id$
 $id - num \bullet * id$
 $id - num * \bullet id$
 $F.2 id - num * \bullet id$
 $id - num * id \bullet$

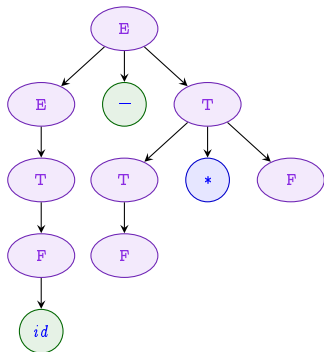


Ülalt-alla parsimine

Näide:

$E \rightarrow E + T$
 $\quad | E - T$
 $\quad | T$
 $T \rightarrow T * F$
 $\quad | T / F$
 $\quad | F$
 $F \rightarrow (E)$
 $\quad | id$
 $\quad | num$

$\bullet id - num * id$
 $E.2 \bullet id - num * id$
 $E.3 \bullet id - num * id$
 $T.3 \bullet id - num * id$
 $F.2 \bullet id - num * id$
 $id \bullet - num * id$
 $id - \bullet num * id$
 $T.1 id - \bullet num * id$
 $T.3 id - \bullet num * id$
 $F.3 id - \bullet num * id$
 $id - num \bullet * id$
 $id - num * \bullet id$
 $F.2 id - num * \bullet id$
 $id - num * id \bullet$

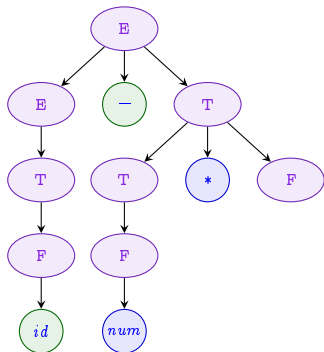


Ülalt-alla parsimine

Näide:

$E \rightarrow E + T$
 $E \rightarrow E - T$
 $E \rightarrow T$
 $T \rightarrow T * F$
 $T \rightarrow T / F$
 $T \rightarrow F$
 $F \rightarrow (E)$
 $F \rightarrow id$
 $F \rightarrow num$

$\bullet id - num * id$
 $E.2 \bullet id - num * id$
 $E.3 \bullet id - num * id$
 $T.3 \bullet id - num * id$
 $F.2 \bullet id - num * id$
 $id \bullet - num * id$
 $id - \bullet num * id$
 $T.1 id - \bullet num * id$
 $T.3 id - \bullet num * id$
 $F.3 id - \bullet num * id$
 $id - num \bullet * id$
 $id - num * \bullet id$
 $F.2 id - num * \bullet id$
 $id - num * id \bullet$

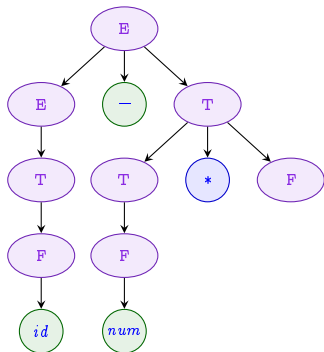


Ülalt-alla parsimine

Näide:

$E \rightarrow E + T$
 $\quad | E - T$
 $\quad | T$
 $T \rightarrow T * F$
 $\quad | T / F$
 $\quad | F$
 $F \rightarrow (E)$
 $\quad | id$
 $\quad | num$

$\bullet id - num * id$
 $E.2 \bullet id - num * id$
 $E.3 \bullet id - num * id$
 $T.3 \bullet id - num * id$
 $F.2 \bullet id - num * id$
 $id \bullet - num * id$
 $id - \bullet num * id$
 $T.1 id - \bullet num * id$
 $T.3 id - \bullet num * id$
 $F.3 id - \bullet num * id$
 $id - num \bullet * id$
 $id - num * \bullet id$
 $F.2 id - num * \bullet id$
 $id - num * id \bullet$

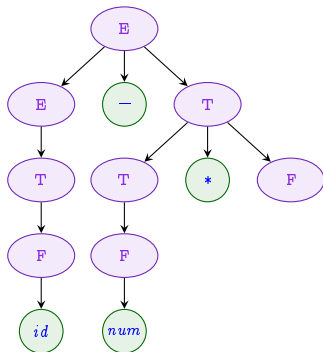


Ülalt-alla parsimine

Näide:

$E \rightarrow E + T$
 $\quad | E - T$
 $\quad | T$
 $T \rightarrow T * F$
 $\quad | T / F$
 $\quad | F$
 $F \rightarrow (E)$
 $\quad | id$
 $\quad | num$

$\bullet id - num * id$
 $E.2 \bullet id - num * id$
 $E.3 \bullet id - num * id$
 $T.3 \bullet id - num * id$
 $F.2 \bullet id - num * id$
 $id \bullet - num * id$
 $id - \bullet num * id$
 $T.1 id - \bullet num * id$
 $T.3 id - \bullet num * id$
 $F.3 id - \bullet num * id$
 $id - num \bullet * id$
 $id - num * \bullet id$
 $F.2 id - num * \bullet id$
 $id - num * id \bullet$

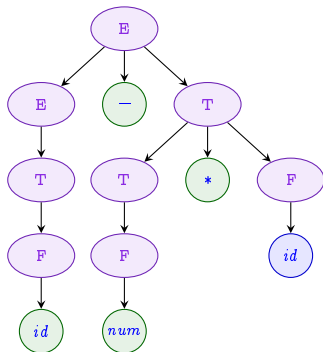


Ülalt-alla parsimine

Näide:

$E \rightarrow E + T$
 $E \rightarrow E - T$
 $E \rightarrow T$
 $T \rightarrow T * F$
 $T \rightarrow T / F$
 $T \rightarrow F$
 $F \rightarrow (E)$
 $F \rightarrow id$
 $F \rightarrow num$

$\bullet id - num * id$
 $E.2 \bullet id - num * id$
 $E.3 \bullet id - num * id$
 $T.3 \bullet id - num * id$
 $F.2 \bullet id - num * id$
 $id \bullet - num * id$
 $id - \bullet num * id$
 $T.1 id - \bullet num * id$
 $T.3 id - \bullet num * id$
 $F.3 id - \bullet num * id$
 $id - num \bullet * id$
 $id - num * \bullet id$
 $F.2 id - num * \bullet id$
 $id - num * id \bullet$

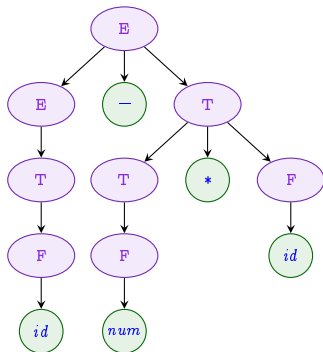


Ülalt-alla parsimine

Näide:

$E \rightarrow E + T$
| $E - T$
| T
 $T \rightarrow T * F$
| T / F
| F
 $F \rightarrow (E)$
| id
| num

• $id - num * id$
 $E.2$ • $id - num * id$
 $E.3$ • $id - num * id$
 $T.3$ • $id - num * id$
 $F.2$ • $id - num * id$
 $id \bullet - num * id$
 $id - \bullet num * id$
 $T.1$ $id - \bullet num * id$
 $T.3$ $id - \bullet num * id$
 $F.3$ $id - \bullet num * id$
 $id - num \bullet * id$
 $id - num * \bullet id$
 $F.2$ $id - num * \bullet id$
 $id - num * id \bullet$



Ülalt-alla parsimine

- Parsimise efektiivsus sõltub tugevasti õige produktsioonireegli valikust.
- Vale reegli valimine põhjustab hilisema tagasipöördumise vajaduse.
- Vasakrekursiivsete reeglite olemasolul võib ülalt-alla parsimine **mittetermineeruda**.

• *id - num * id*

E.2 • *id - num * id*

E.2 • *id - num * id*

E.2 • *id - num * id*

...

Ülalt-alla parsimine

- Parsimise efektiivsus sõltub tugevasti õige produktsioonireegli valikust.
- Vale reegli valimine põhjustab hilisema tagasipöördumise vajaduse.
- Vasakrekursiivsete reeglite olemasolul võib ülalt-alla parsimine **mittetermineeruda**.

E

- *id - num * id*

E.2 • *id - num * id*

E.2 • *id - num * id*

E.2 • *id - num * id*

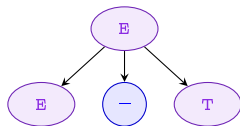
...

Ülalt-alla parsimine

- Parsimise efektiivsus sõltub tugevasti õige produktsioonireegli valikust.
- Vale reegli valimine põhjustab hilisema tagasipöördumise vajaduse.
- Vasakrekursiivsete reeglite olemasolul võib ülalt-alla parsimine **mittetermineeruda**.

- $id - num * id$
- E.2* ● $id - num * id$
- E.2* ● $id - num * id$
- E.2* ● $id - num * id$

...

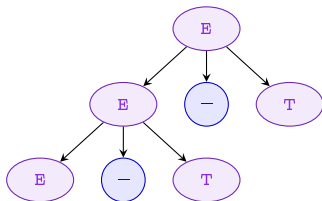


Ülalt-alla parsimine

- Parsimise efektiivsus sõltub tugevasti õige produktsioonireegli valikust.
- Vale reegli valimine põhjustab hilisema tagasipöördumise vajaduse.
- Vasakrekursiivsete reeglite olemasolul võib ülalt-alla parsimine **mittetermineeruda**.

- $id - num * id$
- E.2* • $id - num * id$
- E.2* • $id - num * id$
- E.2* • $id - num * id$

...

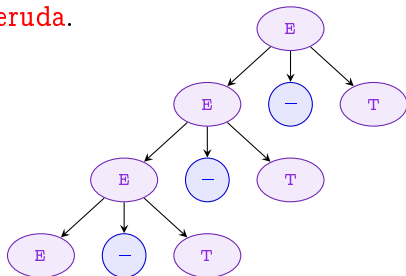


Ülalt-alla parsimine

- Parsimise efektiivsus sõltub tugevasti õige produktsioonireegli valikust.
- Vale reegli valimine põhjustab hilisema tagasipöördumise vajaduse.
- Vasakrekursiivsete reeglite olemasolul võib ülalt-alla parsimine **mittetermineeruda**.

- $id - num * id$
- E.2* • $id - num * id$
- E.2* • $id - num * id$
- E.2* • $id - num * id$

...

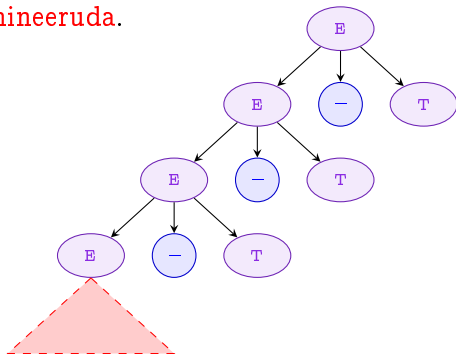


Ülalt-alla parsimine

- Parsimise efektiivsus sõltub tugevasti õige produktsioonireegli valikust.
- Vale reegli valimine põhjustab hilisema tagasipöördumise vajaduse.
- Vasakrekursiivsete reeglite olemasolul võib ülalt-alla parsimine **mittetermineeruda**.

- $id - num * id$
- $E.2$ • $id - num * id$
- $E.2$ • $id - num * id$
- $E.2$ • $id - num * id$

...



Vasakrekursioon

- Grammatika on **vasakrekursiivne**, kui leidub mitteterminal $A \in N$ selline, et

$$A \Longrightarrow^+ A\alpha,$$

kus $\alpha \in V^*$.

- Vasakrekursioon on **vahetu**, kui leidub reegel kujul $A \rightarrow A\alpha$.
- Vastasel korral on vasakrekursioon **kaudne**.

Vasakrekursiooni elimineerimine

Vahetu vasakrekursiooni eemaldamine:

- Toome sisse uue mitteterminali ning asendame vasakrekursiooni paremrekursiooniga

$$A \rightarrow A \alpha \mid \beta \quad \longrightarrow \quad \begin{array}{l} A \rightarrow \beta A' \\ A' \rightarrow \alpha A' \mid \varepsilon \end{array}$$

- Üldjuhul

$$A \rightarrow A \alpha_1 \mid A \alpha_2 \mid \dots \mid \beta_1 \mid \beta_2 \mid \dots$$
$$\longrightarrow \quad \begin{array}{l} A \rightarrow \beta_1 A' \mid \beta_2 A' \mid \dots \\ A' \rightarrow \alpha_2 A' \mid \alpha_2 A' \mid \dots \mid \varepsilon \end{array}$$

Vasakrekursiooni elimineerimine

Näide:

$$\begin{array}{l} E \rightarrow E + T \\ \quad | \quad E - T \\ \quad | \quad T \end{array}$$



$$\begin{array}{l} E \rightarrow T E' \\ E' \rightarrow + T E' \\ \quad | \quad - T E' \\ \quad | \quad \varepsilon \end{array}$$

$$\begin{array}{l} T \rightarrow T * F \\ \quad | \quad T / F \\ \quad | \quad F \end{array}$$



$$\begin{array}{l} T \rightarrow F T' \\ T' \rightarrow * F T' \\ \quad | \quad / F T' \\ \quad | \quad \varepsilon \end{array}$$

- Uus grammatika genereerib sama keele, kuid on natuke vähem intuiitivne.
- Mõlemates grammatikates on operaatorid vasakassotsiatiivsed.

Vasakrekursiooni elimineerimine

Kaudse vasakrekursiooni eemaldamine:

- Näide:

$$\begin{aligned}A_1 &\rightarrow A_2 \alpha \mid \beta \\A_2 &\rightarrow A_1 \gamma \mid A_2 \delta\end{aligned}$$

- Teisendame kaudse vasakrekursiooni vahetuks.
- Asendame mitteterminali A_2 reeglite paremates pooltes kõik A_1 esinemised tema definitsiooniga:

$$\begin{aligned}A_1 &\rightarrow A_2 \alpha \mid \beta \\A_2 &\rightarrow A_2 \alpha \gamma \mid \beta \gamma \mid A_2 \delta\end{aligned}$$

- Eemaldame vahetu vasakrekursiooni:

$$\begin{aligned}A_1 &\rightarrow A_2 \alpha \mid \beta \\A_2 &\rightarrow \beta \gamma A'_2 \\A'_2 &\rightarrow \alpha \gamma A'_2 \mid \delta A'_2 \mid \varepsilon\end{aligned}$$

Vasakrekursiooni elimineerimine

Üldine algoritm kaudse vasakrekursiooni eemaldamiseks:

Järjestame mitteterminaalid mingisse järjekorda A_1, \dots, A_n

for $i \leftarrow 1$ **to** n

for $j \leftarrow 1$ **to** $i - 1$

 Asendame iga produktsiooni kujul $A_i \rightarrow A_j \alpha$

 produktsioonidega $A_i \rightarrow \beta_1 \alpha \mid \beta_2 \alpha \mid \dots \mid \beta_k \alpha$,

 kus $A_j \rightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_k$ on kõik A_j produktsioonid

 Eemaldame vahetu vasakrekursiooni mitteterminali A_i
 produktsioonidest

NB! Eeldab, et algses grammatikas pole ei ε -produktsioone, ega tsükleid (so. $A_i \implies^+ A_i$).

Vasakrekursiooni elimineerimine

Näide:

$$\begin{array}{l} A \rightarrow C \alpha \mid A \alpha \\ B \rightarrow A \beta \mid \gamma \\ C \rightarrow B \delta \mid \varphi \end{array}$$



$$\begin{array}{l} A \rightarrow C \alpha A' \\ B \rightarrow A \beta \mid \gamma \\ C \rightarrow B \delta \mid \varphi \\ A' \rightarrow \alpha A' \mid \varepsilon \end{array}$$



$$\begin{array}{l} A \rightarrow C \alpha A' \\ B \rightarrow C \alpha A' \beta \mid \gamma \\ C \rightarrow B \delta \mid \varphi \\ A' \rightarrow \alpha A' \mid \varepsilon \end{array}$$



$$\begin{array}{l} A \rightarrow C \alpha A' \\ B \rightarrow C \alpha A' \beta \mid \gamma \\ C \rightarrow C \alpha A' \beta \delta \mid \gamma \delta \mid \varphi \\ A' \rightarrow \alpha A' \mid \varepsilon \end{array}$$



$$\begin{array}{l} A \rightarrow C \alpha A' \\ B \rightarrow C \alpha A' \beta \mid \gamma \\ C \rightarrow \alpha A' \beta \delta C' \\ A' \rightarrow \alpha A' \mid \varepsilon \\ C' \rightarrow \gamma \delta C' \mid \varphi C' \mid \varepsilon \end{array}$$



$$\begin{array}{l} A \rightarrow C \alpha A' \\ C \rightarrow \alpha A' \beta \delta C' \\ A' \rightarrow \alpha A' \mid \varepsilon \\ C' \rightarrow \gamma \delta C' \mid \varphi C' \mid \varepsilon \end{array}$$

Vasakrekursiooni elimineerimine

Näide:

$$\begin{array}{l}
 A \rightarrow C \alpha \mid A \alpha \\
 B \rightarrow A \beta \mid \gamma \\
 C \rightarrow B \delta \mid \varphi
 \end{array}
 \longrightarrow
 \begin{array}{l}
 A \rightarrow C \alpha \mid A' \\
 B \rightarrow A \beta \mid \gamma \\
 C \rightarrow B \delta \mid \varphi \\
 A' \rightarrow \alpha A' \mid \varepsilon
 \end{array}$$

$$\begin{array}{l}
 A \rightarrow C \alpha \mid A' \\
 B \rightarrow C \alpha \mid A' \beta \mid \gamma \\
 C \rightarrow B \delta \mid \varphi \\
 A' \rightarrow \alpha A' \mid \varepsilon
 \end{array}
 \longrightarrow
 \begin{array}{l}
 A \rightarrow C \alpha \mid A' \\
 B \rightarrow C \alpha \mid A' \beta \mid \gamma \\
 C \rightarrow C \alpha \mid A' \beta \delta \mid \gamma \delta \mid \varphi \\
 A' \rightarrow \alpha A' \mid \varepsilon
 \end{array}$$

$$\begin{array}{l}
 A \rightarrow C \alpha \mid A' \\
 B \rightarrow C \alpha \mid A' \beta \mid \gamma \\
 C \rightarrow \alpha A' \beta \delta \mid C' \\
 A' \rightarrow \alpha A' \mid \varepsilon \\
 C' \rightarrow \gamma \delta \mid C' \mid \varphi \mid C' \mid \varepsilon
 \end{array}
 \longrightarrow
 \begin{array}{l}
 A \rightarrow C \alpha \mid A' \\
 C \rightarrow \alpha A' \beta \delta \mid C' \\
 A' \rightarrow \alpha A' \mid \varepsilon \\
 C' \rightarrow \gamma \delta \mid C' \mid \varphi \mid C' \mid \varepsilon
 \end{array}$$

Vasakrekursiooni elimineerimine

Näide:

$$\begin{array}{l} A \rightarrow C \alpha \mid A \alpha \\ B \rightarrow A \beta \mid \gamma \\ C \rightarrow B \delta \mid \varphi \end{array} \quad \longrightarrow \quad \begin{array}{l} A \rightarrow C \alpha \mid A' \\ B \rightarrow A \beta \mid \gamma \\ C \rightarrow B \delta \mid \varphi \\ A' \rightarrow \alpha A' \mid \varepsilon \end{array}$$

$$\begin{array}{l} A \rightarrow C \alpha \mid A' \\ B \rightarrow C \alpha A' \beta \mid \gamma \\ C \rightarrow B \delta \mid \varphi \\ A' \rightarrow \alpha A' \mid \varepsilon \end{array} \quad \longrightarrow \quad \begin{array}{l} A \rightarrow C \alpha \mid A' \\ B \rightarrow C \alpha A' \beta \mid \gamma \\ C \rightarrow C \alpha A' \beta \delta \mid \gamma \delta \mid \varphi \\ A' \rightarrow \alpha A' \mid \varepsilon \end{array}$$

$$\begin{array}{l} A \rightarrow C \alpha \mid A' \\ B \rightarrow C \alpha A' \beta \mid \gamma \\ C \rightarrow \alpha A' \beta \delta C' \\ A' \rightarrow \alpha A' \mid \varepsilon \\ C' \rightarrow \gamma \delta C' \mid \varphi C' \mid \varepsilon \end{array} \quad \longrightarrow \quad \begin{array}{l} A \rightarrow C \alpha \mid A' \\ C \rightarrow \alpha A' \beta \delta C' \\ A' \rightarrow \alpha A' \mid \varepsilon \\ C' \rightarrow \gamma \delta C' \mid \varphi C' \mid \varepsilon \end{array}$$

Vasakrekursiooni elimineerimine

Näide:

$$\begin{array}{l} A \rightarrow C \alpha \mid A \alpha \\ B \rightarrow A \beta \mid \gamma \\ C \rightarrow B \delta \mid \varphi \end{array}$$



$$\begin{array}{l} A \rightarrow C \alpha \mid A' \\ B \rightarrow A \beta \mid \gamma \\ C \rightarrow B \delta \mid \varphi \\ A' \rightarrow \alpha A' \mid \varepsilon \end{array}$$



$$\begin{array}{l} A \rightarrow C \alpha \mid A' \\ B \rightarrow C \alpha A' \beta \mid \gamma \\ C \rightarrow B \delta \mid \varphi \\ A' \rightarrow \alpha A' \mid \varepsilon \end{array}$$



$$\begin{array}{l} A \rightarrow C \alpha \mid A' \\ B \rightarrow C \alpha A' \beta \mid \gamma \\ C \rightarrow C \alpha A' \beta \delta \mid \gamma \delta \mid \varphi \\ A' \rightarrow \alpha A' \mid \varepsilon \end{array}$$



$$\begin{array}{l} A \rightarrow C \alpha \mid A' \\ B \rightarrow C \alpha A' \beta \mid \gamma \\ C \rightarrow \alpha A' \beta \delta C' \\ A' \rightarrow \alpha A' \mid \varepsilon \\ C' \rightarrow \gamma \delta C' \mid \varphi C' \mid \varepsilon \end{array}$$



$$\begin{array}{l} A \rightarrow C \alpha \mid A' \\ C \rightarrow \alpha A' \beta \delta C' \\ A' \rightarrow \alpha A' \mid \varepsilon \\ C' \rightarrow \gamma \delta C' \mid \varphi C' \mid \varepsilon \end{array}$$

Vasakrekursiooni elimineerimine

Näide:

$$\begin{array}{l}
 A \rightarrow C \alpha \mid A \alpha \\
 B \rightarrow A \beta \mid \gamma \\
 C \rightarrow B \delta \mid \varphi
 \end{array}
 \longrightarrow
 \begin{array}{l}
 A \rightarrow C \alpha \mid A' \\
 B \rightarrow A \beta \mid \gamma \\
 C \rightarrow B \delta \mid \varphi \\
 A' \rightarrow \alpha A' \mid \varepsilon
 \end{array}$$

$$\begin{array}{l}
 A \rightarrow C \alpha \mid A' \\
 B \rightarrow C \alpha A' \beta \mid \gamma \\
 C \rightarrow B \delta \mid \varphi \\
 A' \rightarrow \alpha A' \mid \varepsilon
 \end{array}
 \longrightarrow
 \begin{array}{l}
 A \rightarrow C \alpha \mid A' \\
 B \rightarrow C \alpha A' \beta \mid \gamma \\
 C \rightarrow C \alpha A' \beta \delta \mid \gamma \delta \mid \varphi \\
 A' \rightarrow \alpha A' \mid \varepsilon
 \end{array}$$

$$\begin{array}{l}
 A \rightarrow C \alpha \mid A' \\
 B \rightarrow C \alpha A' \beta \mid \gamma \\
 C \rightarrow \alpha A' \beta \delta C' \\
 A' \rightarrow \alpha A' \mid \varepsilon \\
 C' \rightarrow \gamma \delta C' \mid \varphi C' \mid \varepsilon
 \end{array}
 \longrightarrow
 \begin{array}{l}
 A \rightarrow C \alpha \mid A' \\
 C \rightarrow \alpha A' \beta \delta C' \\
 A' \rightarrow \alpha A' \mid \varepsilon \\
 C' \rightarrow \gamma \delta C' \mid \varphi C' \mid \varepsilon
 \end{array}$$

Vasakrekursiooni elimineerimine

Näide:

$$\begin{array}{l}
 A \rightarrow C \alpha \mid A \alpha \\
 B \rightarrow A \beta \mid \gamma \\
 C \rightarrow B \delta \mid \varphi
 \end{array}
 \longrightarrow
 \begin{array}{l}
 A \rightarrow C \alpha \mid A' \\
 B \rightarrow A \beta \mid \gamma \\
 C \rightarrow B \delta \mid \varphi \\
 A' \rightarrow \alpha A' \mid \varepsilon
 \end{array}$$

$$\begin{array}{l}
 A \rightarrow C \alpha \mid A' \\
 B \rightarrow C \alpha A' \beta \mid \gamma \\
 C \rightarrow B \delta \mid \varphi \\
 A' \rightarrow \alpha A' \mid \varepsilon
 \end{array}
 \longrightarrow
 \begin{array}{l}
 A \rightarrow C \alpha \mid A' \\
 B \rightarrow C \alpha A' \beta \mid \gamma \\
 C \rightarrow C \alpha A' \beta \delta \mid \gamma \delta \mid \varphi \\
 A' \rightarrow \alpha A' \mid \varepsilon
 \end{array}$$

$$\begin{array}{l}
 A \rightarrow C \alpha \mid A' \\
 B \rightarrow C \alpha A' \beta \mid \gamma \\
 C \rightarrow \alpha A' \beta \delta C' \\
 A' \rightarrow \alpha A' \mid \varepsilon \\
 C' \rightarrow \gamma \delta C' \mid \varphi C' \mid \varepsilon
 \end{array}
 \longrightarrow
 \begin{array}{l}
 A \rightarrow C \alpha \mid A' \\
 C \rightarrow \alpha A' \beta \delta C' \\
 A' \rightarrow \alpha A' \mid \varepsilon \\
 C' \rightarrow \gamma \delta C' \mid \varphi C' \mid \varepsilon
 \end{array}$$

Ennustav parsimine

- Vale produktsioonireegli valimine põhjustab tagasipöördumise.
- Tihti on võimalik valitava reegli õigsuse üle otsustada mingi arvu sisendsümbolite ettevaatamise teel.
- Üldjuhul on tarvis ette vaadata piiramata arv sümboleid.
 - Näit.: Cocke-Younger-Kasami või Earley algoritmid.
- **Ennustav parsimine** (**predictive parsing**) on ülalt-alla parsimine, kus alati on võimalik ige reegel valida nii, et pole vaja tagasipöörduda.
 - Grammatika peab olema selline, et järgmine sisendsümbol (või mingi fikseeritud arv sümboleid) määrab unikaalselt ära valitava reegli.

Ennustav parsimine

Iga lausevormi $\alpha \in (N \cup T)^*$ jaoks defineerime:

$$\begin{aligned} \text{Nullable}(\alpha) &= \begin{cases} \text{true} & \text{if } \alpha \Longrightarrow^* \varepsilon \\ \text{false} & \text{otherwise} \end{cases} \\ \text{First}(\alpha) &= \{c \in T \mid \alpha \Longrightarrow^* c\beta\} \end{aligned}$$

kus $\beta \in (N \cup T)^*$.

Ennustav parsimine

Iga lausevormi $\alpha \in (N \cup T)^*$ jaoks defineerime:

$$\begin{aligned} \text{Nullable}(\alpha) &= \begin{cases} \text{true} & \text{if } \alpha \Longrightarrow^* \varepsilon \\ \text{false} & \text{otherwise} \end{cases} \\ \text{First}(\alpha) &= \{c \in T \mid \alpha \Longrightarrow^* c\beta\} \end{aligned}$$

kus $\beta \in (N \cup T)^*$.

Nullable võrrandid:

$$\text{Nullable}(\varepsilon) = \text{true}$$

$$\text{Nullable}(c) = \text{false}$$

$$\text{Nullable}(\alpha\beta) = \text{Nullable}(\alpha) \wedge \text{Nullable}(\beta)$$

$$\text{Nullable}(A) = \text{Nullable}(\alpha_1) \vee \dots \vee \text{Nullable}(\alpha_n)$$

kasutades kõiki produktsioone $A \rightarrow \alpha_i$

Ennustav parsimine

Iga lausevormi $\alpha \in (N \cup T)^*$ jaoks defineerime:

$$\begin{aligned} \text{Nullable}(\alpha) &= \begin{cases} \text{true} & \text{if } \alpha \Longrightarrow^* \varepsilon \\ \text{false} & \text{otherwise} \end{cases} \\ \text{First}(\alpha) &= \{c \in T \mid \alpha \Longrightarrow^* c\beta\} \end{aligned}$$

kus $\beta \in (N \cup T)^*$.

First võrrandid:

$$\begin{aligned} \text{First}(\varepsilon) &= \emptyset \\ \text{First}(c) &= c \\ \text{First}(\alpha\beta) &= \begin{cases} \text{First}(\alpha) \cup \text{First}(\beta) & \text{if } \text{Nullable}(\alpha) \\ \text{First}(\alpha) & \text{otherwise} \end{cases} \\ \text{First}(A) &= \text{First}(\alpha_1) \cup \dots \cup \text{First}(\alpha_n) \\ &\text{kasutades kõiki produktsioone } A \rightarrow \alpha_i \end{aligned}$$

Ennustav parsimine

- Kui grammatikas on reeglid $A \rightarrow \alpha$ ja $A \rightarrow \beta$ sellised, et $First(\alpha) \cap First(\beta) = \emptyset$, siis on esimese sisendsümboli põhjal võimalik otsustada kumba reeglit valida.
- Kehtib ainult juhul, kui $Nullable(\alpha) \vee Nullable(\beta) = false$.
- Vastasel korral vaja vaadata "järgnevat" sümbolit ...

Ennustav parsimine

Lisame uue algsümboli S' ja reegli $S' \rightarrow S\$$, kus $\$$ on spetsiaalne sisendilõpu marker.

Iga mitteterminali $A \in N$ jaoks defineerime hulga:

$$\text{Follow}(A) = \{c \in T \mid S' \Longrightarrow^* \alpha A c \beta\}$$

kus $\alpha, \beta \in (N \cup T)^*$.

Ennustav parsimine

Lisame uue algsümboli S' ja reegli $S' \rightarrow S\$$, kus $\$$ on spetsiaalne sisendilõpu marker.

Iga mitteterminali $A \in N$ jaoks defineerime hulga:

$$\text{Follow}(A) = \{c \in T \mid S' \Longrightarrow^* \alpha A c \beta\}$$

kus $\alpha, \beta \in (N \cup T)^*$.

Follow võrratused:

- iga $A \in N$ jaoks vaatleme kõiki reegleid $B \rightarrow \alpha A \beta$, kus A on parempooles;
- $\text{First}(\beta) \subseteq \text{Follow}(A)$;
- $\text{Follow}(B) \subseteq \text{Follow}(A)$, kui $\text{Nullable}(\beta)$.

Ennustav parsimine

Näide:

$$S \rightarrow ABC$$

$$A \rightarrow aA \mid \varepsilon$$

$$B \rightarrow b \mid \varepsilon$$

$$C \rightarrow c \mid d$$

$$\textit{First}(C) = \{c, d\}$$

$$\textit{First}(B) = \{b\}$$

$$\textit{First}(A) = \{a\}$$

$$\textit{First}(S) = \textit{First}(ABC)$$

$$= \textit{First}(A)$$

$$\cup \textit{First}(B)$$

$$\cup \textit{First}(C)$$

$$= \{a, b, c, d\}$$

$$\textit{Follow}(S) = \{\$ \}$$

$$\textit{Follow}(C) = \{\$ \}$$

$$\textit{Follow}(B) = \textit{First}(C)$$

$$= \{c, d\}$$

$$\textit{Follow}(A) = \textit{First}(B)$$

$$\cup \textit{First}(C)$$

$$= \{b, c, d\}$$

Ennustav parsimine

Iga reegli $A \rightarrow \alpha$ jaoks defineerime **ettevaatamishulga** (**look-ahead set**):

$$\text{la}(A \rightarrow \alpha) = \begin{cases} \text{First}(\alpha) \cup \text{Follow}(A), & \text{if } \text{Nullable}(\alpha) \\ \text{First}(\alpha), & \text{otherwise} \end{cases}$$

Grammatika on **LL(1)**, kui iga (paarikaupa erineva) produktsioonireegli $A \rightarrow \alpha$ ja $A \rightarrow \beta$ korral

$$\text{la}(A \rightarrow \alpha) \cap \text{la}(A \rightarrow \beta) = \emptyset$$

Ennustav parsimine

- Paljusid grammatikaid on võimalik teisendada LL(1) kujule kasutades:
 - vasakrekursiooni elimineerimist;
 - vasakfaktoriseerimist;
 - halvimal juhul üldistame natuke grammatikat (ning kontrollime eemaldatud kitsendusi pärast parsimist).
- **Vasakfaktoriseerimine** (left-factoring) asendab ühise prefiksiga reeglid uute reeglitega, kus ühine prefiks on ainult ühes parempooles.
- Näide:

$$\begin{array}{l} A \rightarrow B a C D \\ \quad | \\ \quad B a C E \end{array} \quad \longrightarrow \quad \begin{array}{l} A \rightarrow B a C Z \\ Z \rightarrow D \\ \quad | \\ \quad E \end{array}$$

Ennustav parsimine

Vasakfaktoriseerimise algoritm:

- 1 Iga mitteterminali $A \in N$ korral leiame pikima prefixi α , mis esineb kahes või rohkemas A produktsioonireegli parempooles.
- 2 Kui $\alpha \neq \varepsilon$, siis asendame kõik A produktsioonid

$$A \rightarrow \alpha \beta_1 \mid \alpha \beta_2 \mid \dots \mid \alpha \beta_n \mid \gamma$$

produktsioonidega

$$\begin{aligned} A &\rightarrow \alpha Z \mid \gamma \\ Z &\rightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_n \end{aligned}$$

kus $Z \in N$ on uus mitteterminaal.

- 3 Kordame protsessi, kuni ükski parempool ei oma ühist prefixit.

Ennustav parsimine

- **Rekursiivselt laskuv** (**recursive descent**) parsimine on ennustava parsimise meetod, kus:
 - iga mitteterminali kohta realiseeritakse üks protseduur, mis tunneb ära sellele mitteterminalile vastavad lausevormid;
 - iga protseduur valib sõltuvalt sisendist reegli ning kutsub (rekursiivselt) välja reegli paremas pooles asuvatele mitteterminalidele vastavad protseduurid.
- Rekursiivselt laskuv parsimine on levinuim meetod (lihtsate keelte) parserite käsitsi kirjutamiseks.

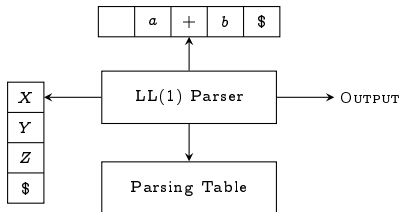
Ennustav parsimine

Näide: olgu A produktsioonireeglid $A \rightarrow a B \mid b C A \mid D E$

```
parseA() {  
  if token = a then {  
    token := nextWord(); parseB();  
  } else if token = b then {  
    token := nextWord(); parseC(); parseA();  
  } else {  
    parseD(); parseE();  
  }  
}
```

Ennustav parsimine

- Automaatselt genereeritavate LL(1) parserite korral kasutatakse tavaliselt tabeljuhitavat LL(1) parsimist:
 - moodustatakse maatriks M , kus read ja veerud on vastavalt indekseeritud mitteterminalide ja terminalidega;
 - tabeli pesades on produktsioonireglid, mida antud mitteterminali ja sisendsümboli korral valida.
- Tabeljuhitava LL(1) parseri struktuur:



Ennustav parsimine

LL(1) parsimise algoritm:

```
push($); push(S);  
token := nextWord();  
while stack ≠ empty do {  
    A := pop();  
    if  $A \in N$  then {  
        if  $M[A, token] = B_1 \dots B_n$  then {  
            push( $B_n$ ); ...; push( $B_1$ );  
        } else error();  
    } else if  $A = token$  then {  
        token := nextWord();  
    } else error();  
}
```

Shift-reduce parsimine

Shift-reduce parsimine on üldine meetod alt-üles süntaksanalüüsiks:

- puu konstrueerimine toimub lehtedest juure suunas eesmärgiga "taandada" sisendstring algsümboliks;
- parsimise ajal on korraga terve mets puid, mis vastavad erinevatele, juba äratuntud, alamstringidele;
- kaks baasaktsiooni:
 - **shift** loeb uue sisendsümboli;
 - **reduce** taandab mingi produktsioonireegli parempoolele vastava (juba loetud/taandatud sisendsümbolite ja mitteterminalide) jada reegli vasakpoolele olevaks mitteterminaliks;
- konstruktsioon vastab parenderivatsioonile.

Shift-reduce parsimone

Näide:

Input String: • *abccde*

Stack:

S → *a A B e*

A → *b c A* | *c*

B → *d*

shift

shift

shift

shift

reduce *A* → *c*

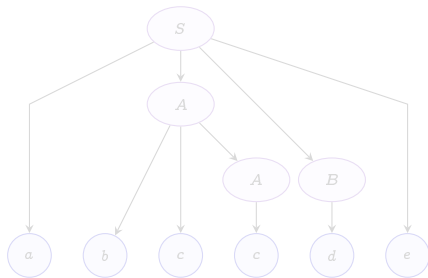
reduce *A* → *b c A*

shift

reduce *B* → *d*

shift

reduce *S* → *a A B e*



$S \Rightarrow_{rm} a A B e \Rightarrow_{rm} a A d e$
 $\Rightarrow_{rm} a b c A d e \Rightarrow_{rm} a b c c d e$

Shift-reduce parsimone

Näide:

Input String: $a \bullet b c c d e$

Stack: a

$S \rightarrow a A B e$

$A \rightarrow b c A \mid c$

$B \rightarrow d$

shift

shift

shift

shift

reduce $A \rightarrow c$

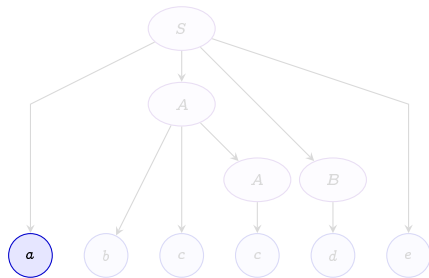
reduce $A \rightarrow b c A$

shift

reduce $B \rightarrow d$

shift

reduce $S \rightarrow a A B e$



$S \xRightarrow{rm} a A B e \xRightarrow{rm} a A d e$
 $\xRightarrow{rm} a b c A d e \xRightarrow{rm} a b c c d e$

Shift-reduce parsimone

Näide:

Input String: $ab \bullet ccde$

Stack: ab

$S \rightarrow aABe$

$A \rightarrow bcA \mid c$

$B \rightarrow d$

shift

shift

shift

shift

reduce $A \rightarrow c$

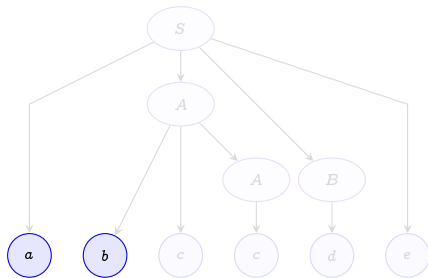
reduce $A \rightarrow bcA$

shift

reduce $B \rightarrow d$

shift

reduce $S \rightarrow aABe$



$S \xRightarrow{rm} aABe \xRightarrow{rm} aAde$
 $\xRightarrow{rm} abcAde \xRightarrow{rm} abccde$

Shift-reduce parsimone

Näide:

Input String: $abc \bullet cde$

Stack: abc

$S \rightarrow aABe$

$A \rightarrow bcA \mid c$

$B \rightarrow d$

shift

shift

shift

shift

reduce $A \rightarrow c$

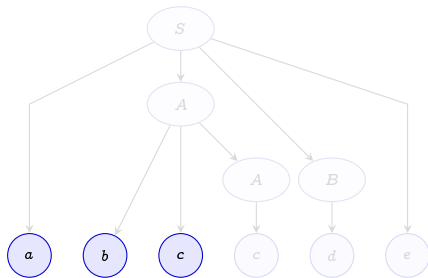
reduce $A \rightarrow bcA$

shift

reduce $B \rightarrow d$

shift

reduce $S \rightarrow aABe$



$S \xRightarrow{rm} aABe \xRightarrow{rm} aAde$
 $\xRightarrow{rm} abcAde \xRightarrow{rm} abccde$

Shift-reduce parsimone

Näide:

Input String: $abcc \bullet de$

Stack: $abcc$

$S \rightarrow a A B e$

$A \rightarrow bcA \mid c$

$B \rightarrow d$

shift

shift

shift

shift

reduce $A \rightarrow c$

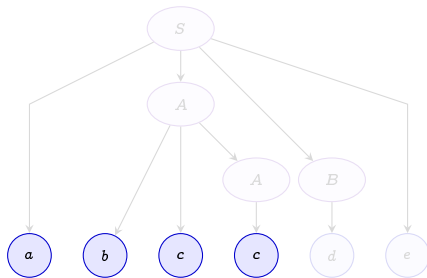
reduce $A \rightarrow bcA$

shift

reduce $B \rightarrow d$

shift

reduce $S \rightarrow a A B e$



$S \xRightarrow{rm} a A B e \xRightarrow{rm} a A d e$
 $\xRightarrow{rm} a b c A d e \xRightarrow{rm} a b c c d e$

Shift-reduce parsimone

Näide:

Input String: $abcc \bullet de$

Stack: $abcA$

$S \rightarrow aABe$

$A \rightarrow bcA \mid c$

$B \rightarrow d$

shift

shift

shift

shift

reduce $A \rightarrow c$

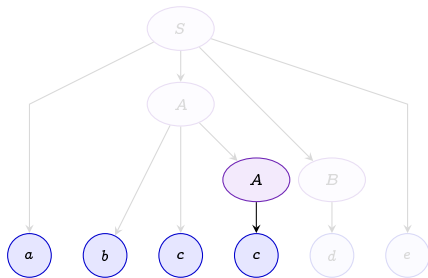
reduce $A \rightarrow bcA$

shift

reduce $B \rightarrow d$

shift

reduce $S \rightarrow aABe$



$S \xRightarrow{rm} aABe \xRightarrow{rm} aAde$
 $\xRightarrow{rm} abcAde \xRightarrow{rm} abccde$

Shift-reduce parsimone

Näide:

Input String: $abcc \bullet de$

Stack: aA

$S \rightarrow aABe$

$A \rightarrow bcA \mid c$

$B \rightarrow d$

shift

shift

shift

shift

reduce $A \rightarrow c$

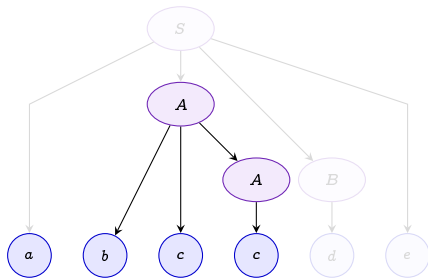
reduce $A \rightarrow bcA$

shift

reduce $B \rightarrow d$

shift

reduce $S \rightarrow aABe$



$S \xRightarrow{rm} aABe \xRightarrow{rm} aAde$
 $\xRightarrow{rm} abccde$

Shift-reduce parsimone

Näide:

Input String: $abccde \bullet e$

Stack: aAd

$S \rightarrow aABe$

$A \rightarrow bcA \mid c$

$B \rightarrow d$

shift

shift

shift

shift

reduce $A \rightarrow c$

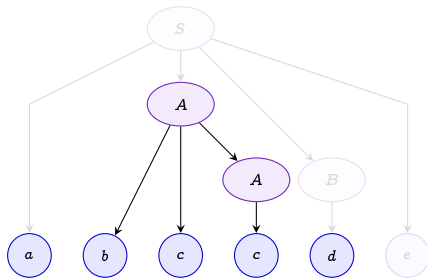
reduce $A \rightarrow bcA$

shift

reduce $B \rightarrow d$

shift

reduce $S \rightarrow aABe$



$S \xRightarrow{rm} aABe \xRightarrow{rm} aAde$
 $\xRightarrow{rm} abcAde \xRightarrow{rm} abccde$

Shift-reduce parsimone

Näide:

Input String: $abc d \bullet e$

Stack: $a A B$

$S \rightarrow a A B e$

$A \rightarrow b c A \mid c$

$B \rightarrow d$

shift

shift

shift

shift

reduce $A \rightarrow c$

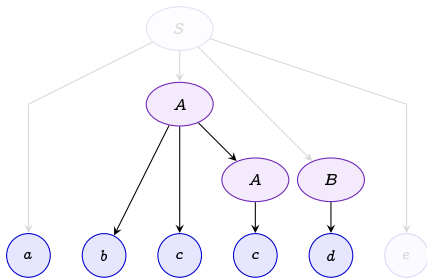
reduce $A \rightarrow b c A$

shift

reduce $B \rightarrow d$

shift

reduce $S \rightarrow a A B e$



$S \Rightarrow_{rm} a A B e \Rightarrow_{rm} a A d e$
 $\Rightarrow_{rm} a b c A d e \Rightarrow_{rm} a b c c d e$

Shift-reduce parsimone

Näide:

Input String: $abcde \bullet$

Stack: $aABe$

$S \rightarrow aABe$

$A \rightarrow bcA \mid c$

$B \rightarrow d$

shift

shift

shift

shift

reduce $A \rightarrow c$

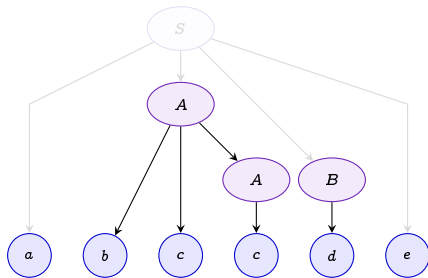
reduce $A \rightarrow bcA$

shift

reduce $B \rightarrow d$

shift

reduce $S \rightarrow aABe$



$S \xRightarrow{rm} aABe \xRightarrow{rm} aAde$
 $\xRightarrow{rm} abcAde \xRightarrow{rm} abccde$

Shift-reduce parsimone

Näide:

Input String: $abcde \bullet$

Stack: S

$S \rightarrow a A B e$

$A \rightarrow bcA \mid c$

$B \rightarrow d$

shift

shift

shift

shift

reduce $A \rightarrow c$

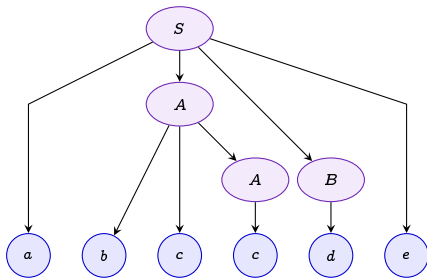
reduce $A \rightarrow bcA$

shift

reduce $B \rightarrow d$

shift

reduce $S \rightarrow a A B e$



$S \Rightarrow_{rm} a A B e \Rightarrow_{rm} a A d e$
 $\Rightarrow_{rm} a b c A d e \Rightarrow_{rm} a b c c d e$

Shift-reduce parsimone

Näide:

$S \rightarrow a A B e$

$A \rightarrow b c A \mid c$

$B \rightarrow d$

shift

shift

shift

shift

reduce $A \rightarrow c$

reduce $A \rightarrow b c A$

shift

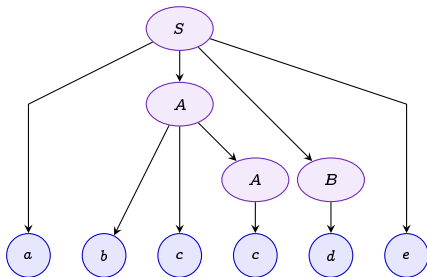
reduce $B \rightarrow d$

shift

reduce $S \rightarrow a A B e$

Input String:

Stack:



$S \xRightarrow{rm} a A B e \xRightarrow{rm} a A d e$
 $\xRightarrow{rm} a b c A d e \xRightarrow{rm} a b c c d e$

Shift-reduce parsimine

- Lausevormi nimetatakse **paremlausevormiks** (**right-sentential form**), kui ta on tuletatav paremderivatsiooni abil.
- Paremlausevormi γ **pide** (**handle**) on produktsioonireegel $A \rightarrow \beta$ ja sõnas γ alamssõna β esinemise positsioon, mis paremderivatsiooni eelmises lausevormis on asendatud mitteterminaliga A .
- Ekvivalentset, paremlausevormi γ pide on alamstring β , selline et $S \xRightarrow{*}_{rm} \delta A w \xRightarrow{rm} \delta \beta w = \gamma$, kus $\beta, \gamma, \delta \in V^*$ ja $w \in T^*$.
- Pidemete leidmise ja nende taandamise protsessi nimetatakse "**pidemete kärpimiseks**" ("**handle pruning**").
- **NB!** Ühese grammatika korral on paremderivatsioon, ja seega ka pidemed, unikaalsed.

Shift-reduce parsimine

Näide: olgu antud grammatika

$$\begin{aligned} S &\rightarrow a A B e \\ A &\rightarrow b c A \mid c \\ B &\rightarrow d \end{aligned}$$

ja parenderivatsioon

$$S \Longrightarrow_{rm} a A B e \Longrightarrow_{rm} a A d e \Longrightarrow_{rm} a b c A d e$$

Paremlausevormi $abcAde$ pide on bcA .

Shift-reduce parsimine

- Shift-reduce parseris tekib pide enne taandamist magasinis tippu.
- Paremlausevormi prefikseid, mis ei lõpe kaugemal paremal kui selles lausevormis olev pide, nimetatakse **elujõulisteks** (**viable**).
- Elujõulised prefiksud on shift-reduce parseri võimalikeks magasinideks!
- **NB!** "Elujõuliste prefiksude keel" on regulaarne!
- Järelikult leidub lõplik automaat, mis tunneb ära elujõulisi prefikseid.
- See automaat on kõigi LR-parserite põhikomponendiks.

Shift-reduce parsimine

- **LR(0)-element** (või lihtsalt **element**; *ingl. item*) on produktioonireegel, mille parempoolde on kuhugi lisatud punkt.
- Näide: olgu antud grammatika

$$\begin{aligned} S &\rightarrow a A c \\ A &\rightarrow A b \mid \varepsilon \end{aligned}$$

Tema LR(0)-elemendid on:

$$\begin{array}{ll} [S \rightarrow \cdot a A c] & [A \rightarrow \cdot A b] \\ [S \rightarrow a \cdot A c] & [A \rightarrow A \cdot b] \\ [S \rightarrow a A \cdot c] & [A \rightarrow A b \cdot] \\ [S \rightarrow a A c \cdot] & [A \rightarrow \cdot] \end{array}$$

Shift-reduce parsimine

- Element $[A \rightarrow \alpha \cdot \beta]$ on elujõulise prefiksi φ suhtes **kehtiv** (**valid**), kui leidub paremderivatsioon

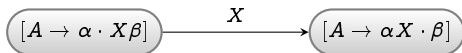
$$S \Longrightarrow_{rm}^* \delta A w \Longrightarrow_{rm} \delta \alpha \beta w$$

ja $\delta \alpha = \varphi$.

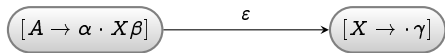
- Kehtiv element $[A \rightarrow \alpha \cdot \beta]$ tähendab, et seni nähtud sisend on kooskõlas reegli $A \rightarrow \alpha \beta$ rakendamisega ja parser on juba ära tundnud α .
- Kehtiv element $[A \rightarrow \alpha \beta \cdot]$ tähendab, et parser on ära tundnud $\alpha \beta$ ja on kooskõlas selle redutseerimisega mitteterminaliks A .

Shift-reduce parsimine

- Mittedetermineeritud LR(0)-automaat on NFA, kus olekuks on elemendid ja kus on kaht liiki üleminekuid:
 - elementide $[A \rightarrow \alpha \cdot X\beta]$ ja $[A \rightarrow \alpha X \cdot \beta]$ vahel on (terminal- või mitteterminal-) sümboliga X märgendatud üleminek



- elementide $[A \rightarrow \alpha \cdot X\beta]$ ja $[X \rightarrow \cdot \gamma]$ vahel on tühisümboliga ϵ märgendatud üleminek



- Grammatika on laiendatud uue algsümboliga S' ja ühe uue reegluga $S' \rightarrow S$.
- Elementi $[S' \rightarrow \cdot S]$ sisaldav olek on automaadi algolekuks.

Shift-reduce parsimone

$S \rightarrow a A B e$

$A \rightarrow b c A \mid c$

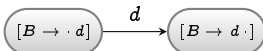
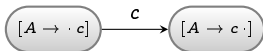
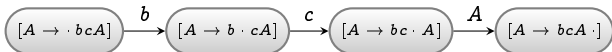
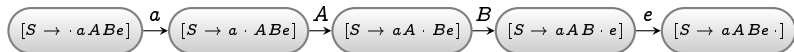
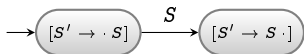
$B \rightarrow d$

Shift-reduce parsimime

$S \rightarrow aABe$

$A \rightarrow bcA \mid c$

$B \rightarrow d$

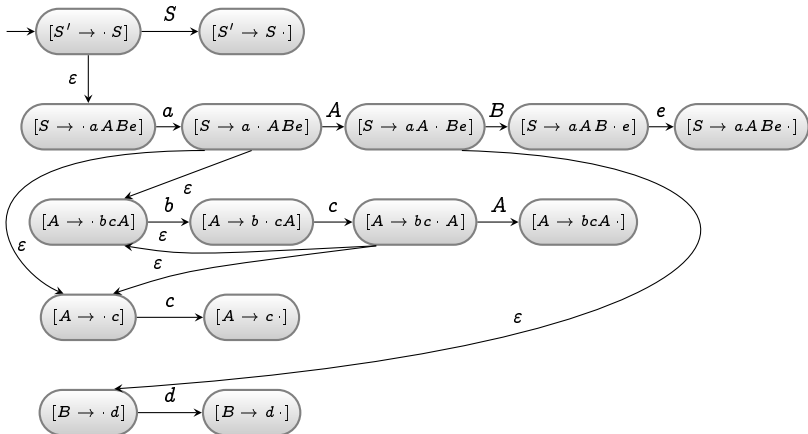


Shift-reduce parsimime

$S \rightarrow aABe$

$A \rightarrow bcA \mid c$

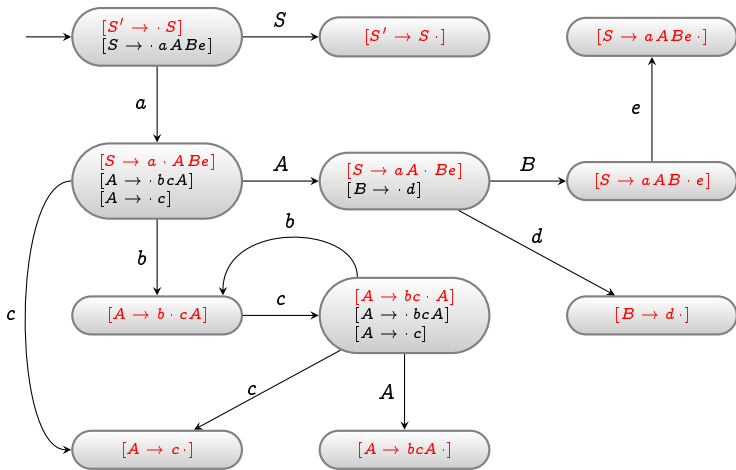
$B \rightarrow d$



Shift-reduce parsimine

- NFA-st saame konstrueerida DFA kasutades osahulkade moodustamise konstruktsiooni.
- DFA olekuteks on elementide hulgad.
- Elemente, mis on mitte- ϵ -üleminekute sihiks, nimetatakse **tuumelementideks** (**kernel items**).
- Elemente, mis on ϵ -üleminekute sihiks, nimetatakse **sulundelementideks** (**closure items**).
- Tuumelemendid identifitseerivad üheselt oleku.

Shift-reduce parsimone



Shift-reduce parsimime

LR(0) parsimialgorithm:

```
push([S' → · S]);
while true do {
  state := top();
  if  $\exists [A \rightarrow \alpha \cdot X \beta] \in state \wedge X \in T$  then
    Y := getToken();
    if  $\exists [A \rightarrow \alpha \cdot Y \beta] \in state$  then          /* shift */
      push([A →  $\alpha Y \cdot \beta$ ]);
    else error;
  else if  $\exists [A \rightarrow \gamma \cdot] \in state$  then
    if A = S'  $\wedge \gamma = S$  then accept;
    else pop(| $\gamma$ |); state := top();          /* reduce */
      if  $\exists [B \rightarrow \alpha \cdot A \beta] \in state$  then
        push([B →  $\alpha A \cdot \beta$ ]);
      else error;
  else error;
}
```

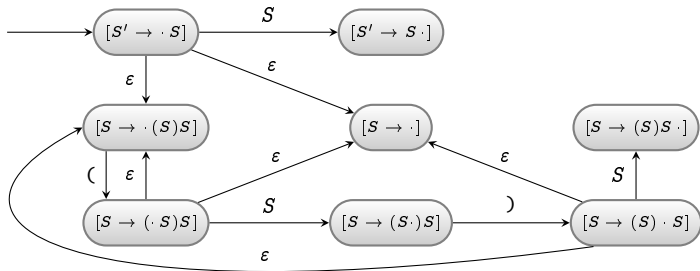
Shift-reduce parsimine

- Grammatika on **LR(0)**, kui iga täiselement (so. element kujul $[A \rightarrow \alpha \cdot] \in state$) on teda sisaldava oleku *state* ainsaks elemendiks.
 - **Shift-reduce konflikt**: kui $\exists [B \rightarrow \alpha \cdot \beta] \in state$;
 - **Reduce-reduce konflikt**: kui $\exists [B \rightarrow \beta \cdot] \in state$.
- **Reduce olekud**: täiselemente sisaldavad olekud.
- **Shift olekud**: kõik ülejäänud olekud.

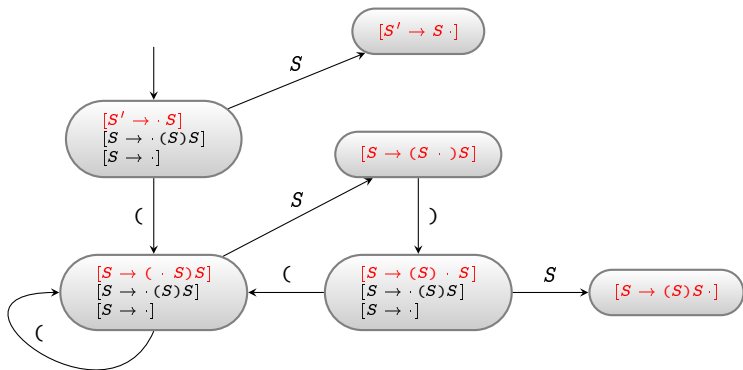
Shift-reduce parsimone

$S' \rightarrow S$

$S \rightarrow (S)S \mid \epsilon$



Shift-reduce parsimone



Shift-reduce parsimine

- **SLR(1)** = **Simple LR(1)** parsimine.
- LR(0) parsimise efektiivne laiendus, mis on piisavalt võimas mitmete praktiliste keelte käsitlemiseks.
- Kasutab LR(0) elementidest moodustatud DFA-d.
- Sarnane LR(0) parsimisele, kuid aktsiooni valiku otsustus on lükatud võimalikult hiliseks.
 - Kontrollib sisendsümbolit enne shift'i, et kindlustada vastava DFA ülemineku olemasolu.
 - Kasutab mitteterminali Follow hulka, et otsustada milline reduktsioon teha.

Shift-reduce parsimime

SLR(1) parsimialgorithm:

```
push([S' → · S]);  
while true do {  
    state := top(); X := getToken();  
    if  $\exists[A \rightarrow \alpha \cdot X \beta] \in state$  then /* shift */  
        push([A →  $\alpha X \cdot \beta$ ]);  
    else if  $\exists[A \rightarrow \gamma \cdot] \in state \wedge X \in Follow(A)$  then  
        if  $A = S' \wedge \gamma = S \wedge X = \$$  then accept;  
        else pop(| $\gamma$ |); state := top(); /* reduce */  
            if  $\exists[B \rightarrow \alpha \cdot A \beta] \in state$  then  
                push([B →  $\alpha A \cdot \beta$ ]);  
            else error;  
    else error;  
}
```

Shift-reduce parsimine

Grammatika on **SLR(1)** kui tema DFA-s ei ole järgnevaid konflikte:

- **Shift-reduce konflikt:**

$$\begin{aligned} \forall [A \rightarrow \alpha \cdot X \beta] \in state \wedge X \in T \\ \Rightarrow \neg (\exists [B \rightarrow \gamma \cdot] \in state \wedge X \in Follow(B)) \end{aligned}$$

- **Reduce-reduce konflikt:**

$$\begin{aligned} \forall [A \rightarrow \alpha \cdot] \in state \wedge [B \rightarrow \beta \cdot] \in state \\ \Rightarrow Follow(A) \cap Follow(B) = \emptyset \end{aligned}$$