Süntaksanalüüs

Shift-reduce parsimine
LR(0) parsimine
SLR(1) parsimine

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 erinavatele, juba äratuntud, alamstringidele;
- kaks baasaktsiooni:
  - shift loeb uue sisendsümboli;
  - reduce taandab mingi produktsioonireegli parempoolele vastava (juba loetud/taandatud sisendsümbolite ja mitteterminalide) jada reegli vasakpoolel olevaks mitteterminaliks;
- konstruktsioon vastab paremderivatsioonile.

Shift-reduce parsimine

Näide:

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

Stack:

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

shift
shift
shift
shift
reduce $A \rightarrow c$
reduce $A \rightarrow b\, c\, A$
shift
reduce $B \rightarrow d$
shift
reduce $S \rightarrow a\, A\, B\, e$

$$
\begin{array}{lllll}
S & \Longrightarrow_{rm} & a\, A\, B\, e & \Longrightarrow_{rm} & a\, A\, d\, e \\
 & \Longrightarrow_{rm} & a\, b\, c\, A\, d\, e & \Longrightarrow_{rm} & a\, b\, c\, c\, d\, e
\end{array}
$$

Shift-reduce parsimine

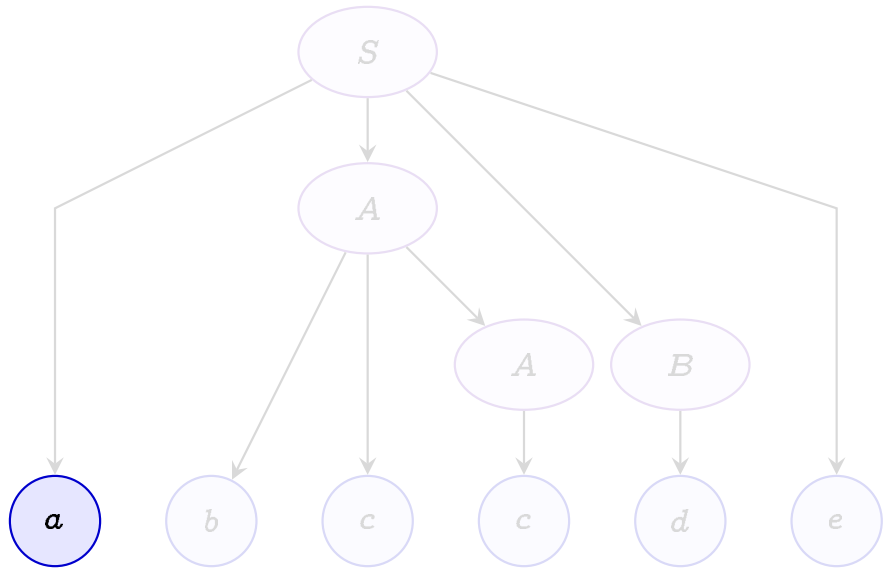
Näide:

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

Stack: a

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

shift
shift
shift
shift
reduce $A \rightarrow c$
reduce $A \rightarrow b\,c\,A$
shift
reduce $B \rightarrow d$
shift
reduce $S \rightarrow a\,A\,B\,e$



$$
\begin{array}{lllll}
S & \Longrightarrow_{rm} & a\,A\,B\,e & \Longrightarrow_{rm} & a\,A\,d\,e \\
 & \Longrightarrow_{rm} & a\,b\,c\,A\,d\,e & \Longrightarrow_{rm} & a\,b\,c\,c\,d\,e
\end{array}
$$

Shift-reduce parsimine

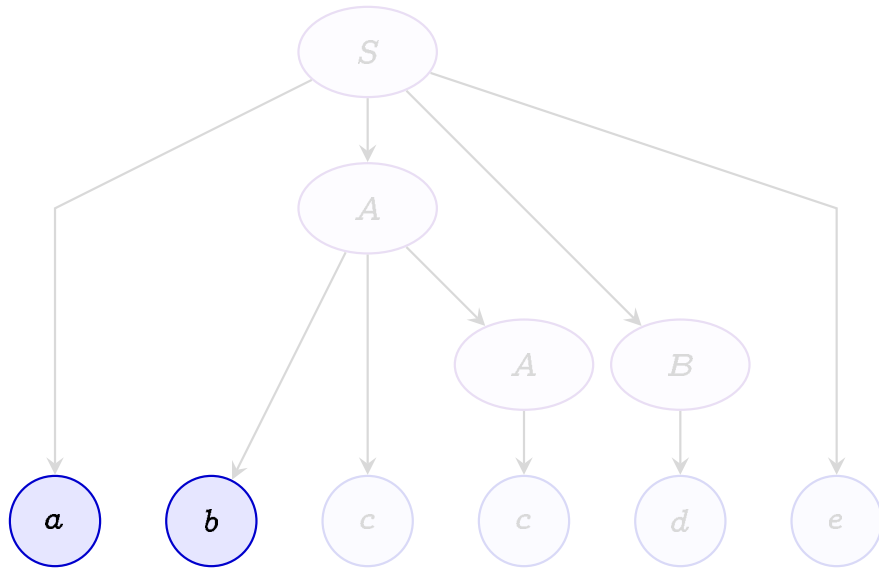
Näide:

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

Stack: $a\,b$

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

shift
shift
shift
shift
reduce $A \rightarrow c$
reduce $A \rightarrow b\,c\,A$
shift
reduce $B \rightarrow d$
shift
reduce $S \rightarrow a\,A\,B\,e$



$$
\begin{array}{lllll}
S & \Longrightarrow_{rm} & a\,A\,B\,e & \Longrightarrow_{rm} & a\,A\,d\,e \\
 & \Longrightarrow_{rm} & a\,b\,c\,A\,d\,e & \Longrightarrow_{rm} & a\,b\,c\,c\,d\,e
\end{array}
$$

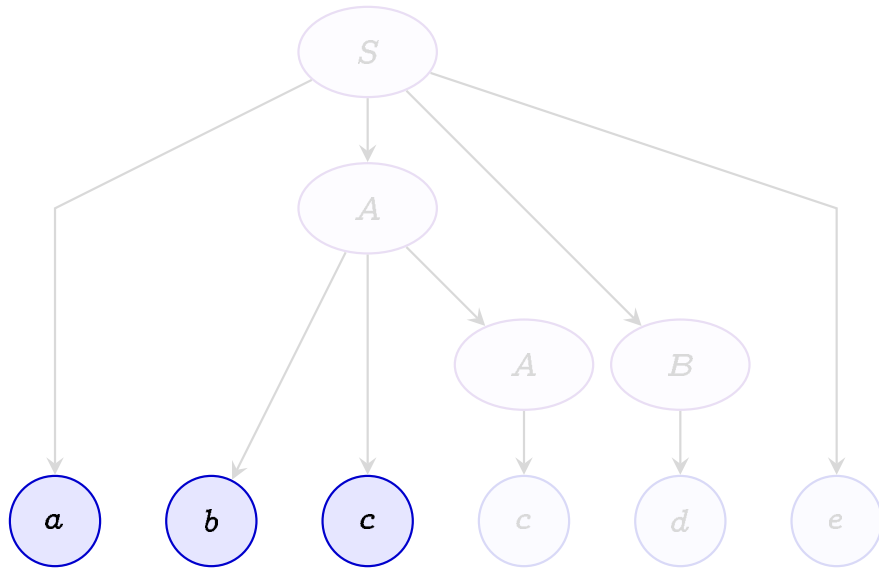
Shift-reduce parsimine

Näide:

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

Stack: $a\,b\,c$

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



shift
shift
shift
shift
reduce $A \rightarrow c$
reduce $A \rightarrow b\,c\,A$
shift
reduce $B \rightarrow d$
shift
reduce $S \rightarrow a\,A\,B\,e$

$$
\begin{array}{lllll}
S & \Longrightarrow_{rm} & a\,A\,B\,e & \Longrightarrow_{rm} & a\,A\,d\,e \\
 & \Longrightarrow_{rm} & a\,b\,c\,A\,d\,e & \Longrightarrow_{rm} & a\,b\,c\,c\,d\,e
\end{array}
$$

Shift-reduce parsimine

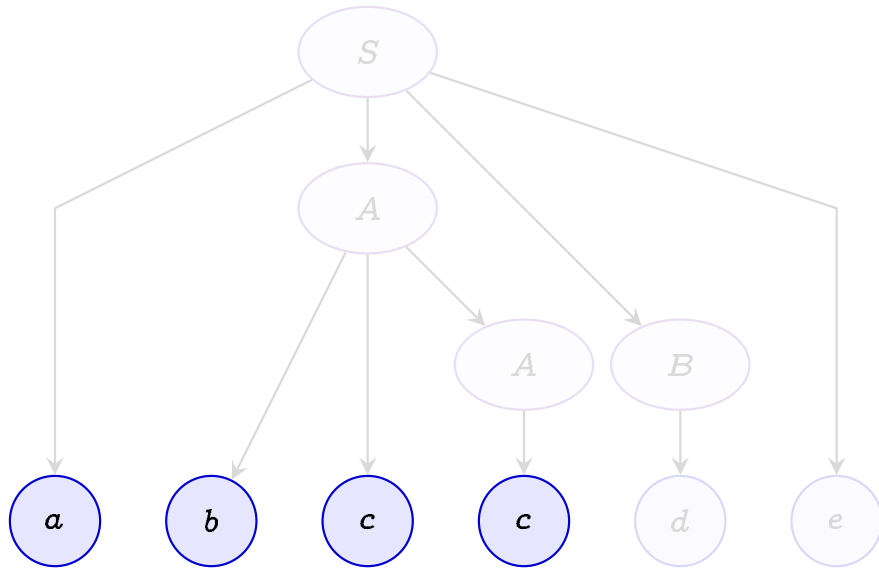
Näide:

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

Stack: $a\,b\,c\,c$

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

shift
shift
shift
shift
reduce $A \rightarrow c$
reduce $A \rightarrow b\,c\,A$
shift
reduce $B \rightarrow d$
shift
reduce $S \rightarrow a\,A\,B\,e$



$$
\begin{array}{lllll}
S & \Longrightarrow_{rm} & a\,A\,B\,e & \Longrightarrow_{rm} & a\,A\,d\,e \\
 & \Longrightarrow_{rm} & a\,b\,c\,A\,d\,e & \Longrightarrow_{rm} & a\,b\,c\,c\,d\,e
\end{array}
$$

Shift-reduce parsimine

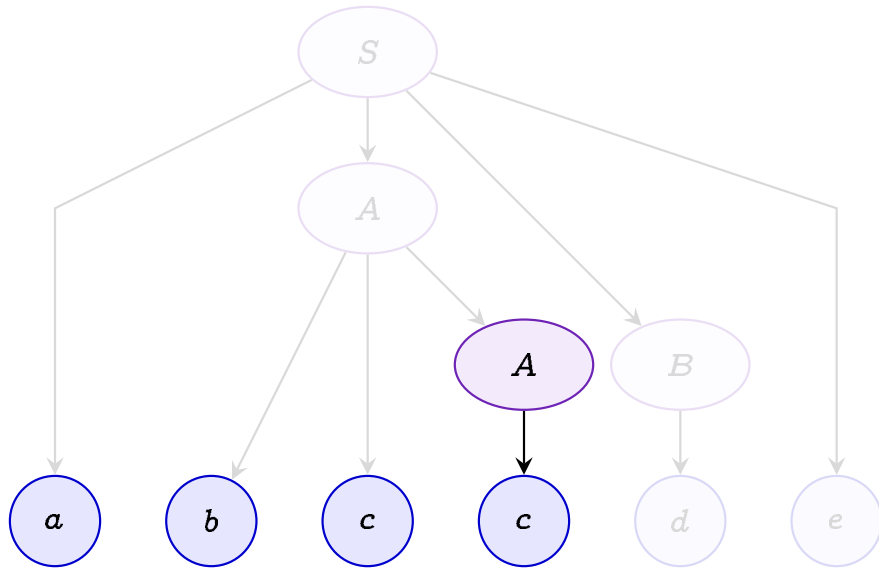
Näide:

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

Stack: $a\,b\,c\,A$

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

shift
shift
shift
shift
reduce $A \rightarrow c$
reduce $A \rightarrow b\,c\,A$
shift
reduce $B \rightarrow d$
shift
reduce $S \rightarrow a\,A\,B\,e$



$$
\begin{array}{lllll}
S & \Longrightarrow_{rm} & a\,A\,B\,e & \Longrightarrow_{rm} & a\,A\,d\,e \\
 & \Longrightarrow_{rm} & a\,b\,c\,A\,d\,e & \Longrightarrow_{rm} & a\,b\,c\,c\,d\,e
\end{array}
$$

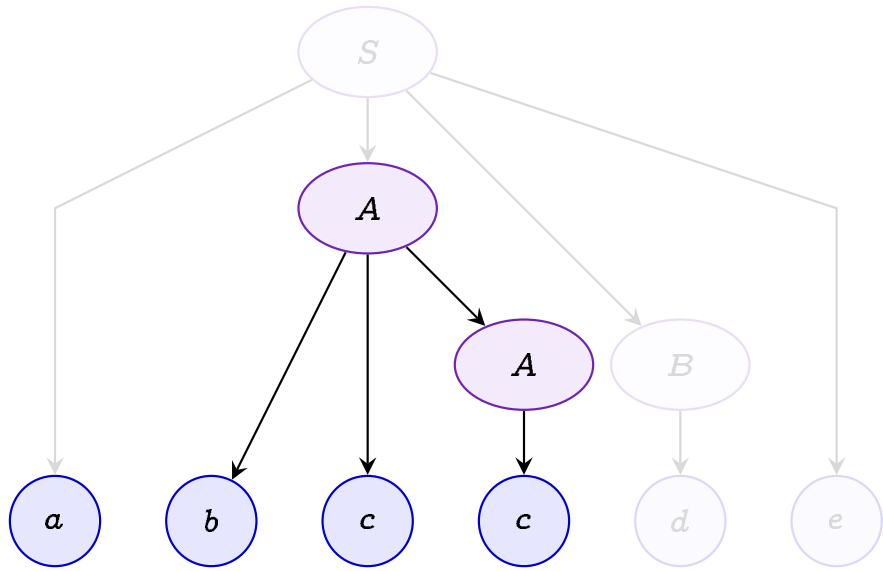
Shift-reduce parsimine

Näide:

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

Stack: $a\,A$

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



shift
shift
shift
shift
reduce $A \rightarrow c$
reduce $A \rightarrow b\,c\,A$
shift
reduce $B \rightarrow d$
shift
reduce $S \rightarrow a\,A\,B\,e$

$$
\begin{array}{lllll}
S & \Longrightarrow_{rm} & a\,A\,B\,e & \Longrightarrow_{rm} & a\,A\,d\,e \\
 & \Longrightarrow_{rm} & a\,b\,c\,A\,d\,e & \Longrightarrow_{rm} & a\,b\,c\,c\,d\,e
\end{array}
$$

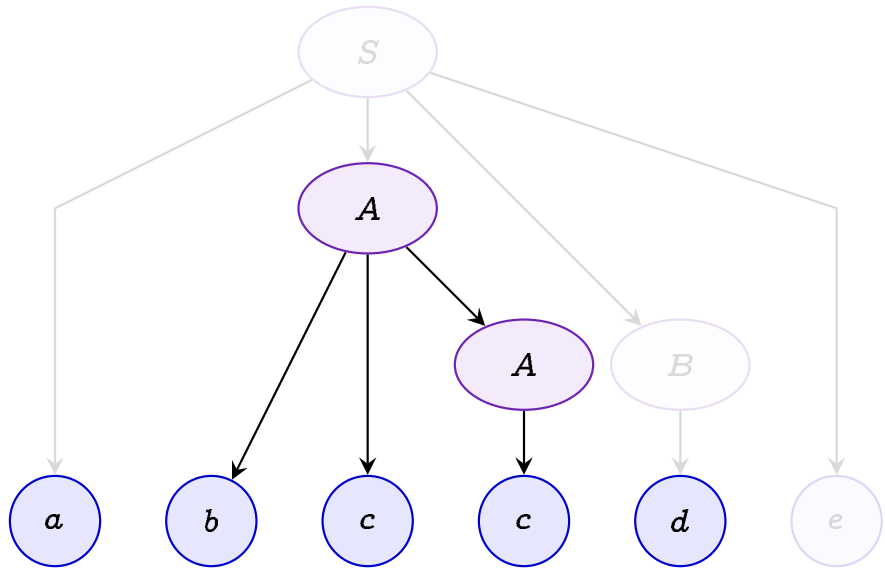
Shift-reduce parsimine

Näide:

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

Stack: $a\,A\,d$

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



shift
shift
shift
shift
reduce $A \rightarrow c$
reduce $A \rightarrow b\,c\,A$
shift
reduce $B \rightarrow d$
shift
reduce $S \rightarrow a\,A\,B\,e$

$$
\begin{array}{lllll}
S & \Longrightarrow_{rm} & a\,A\,B\,e & \Longrightarrow_{rm} & a\,A\,d\,e \\
 & \Longrightarrow_{rm} & a\,b\,c\,A\,d\,e & \Longrightarrow_{rm} & a\,b\,c\,c\,d\,e
\end{array}
$$

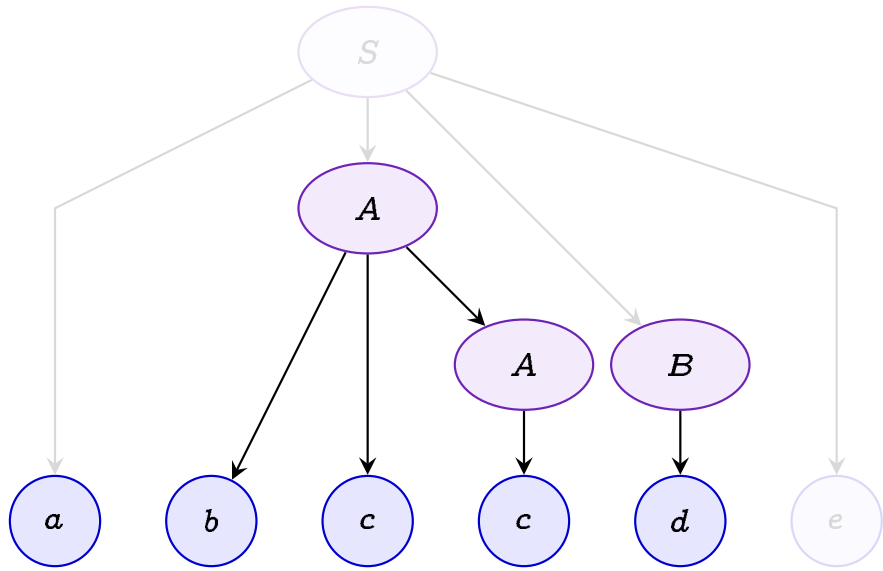
Shift-reduce parsimine

Näide:

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

Stack: $a\,A\,B$

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



shift
shift
shift
shift
reduce $A \rightarrow c$
reduce $A \rightarrow b\,c\,A$
shift
reduce $B \rightarrow d$
shift
reduce $S \rightarrow a\,A\,B\,e$

$$
\begin{array}{lllll}
S & \Longrightarrow_{rm} & a\,A\,B\,e & \Longrightarrow_{rm} & a\,A\,d\,e \\
 & \Longrightarrow_{rm} & a\,b\,c\,A\,d\,e & \Longrightarrow_{rm} & a\,b\,c\,c\,d\,e
\end{array}
$$

Shift-reduce parsimine

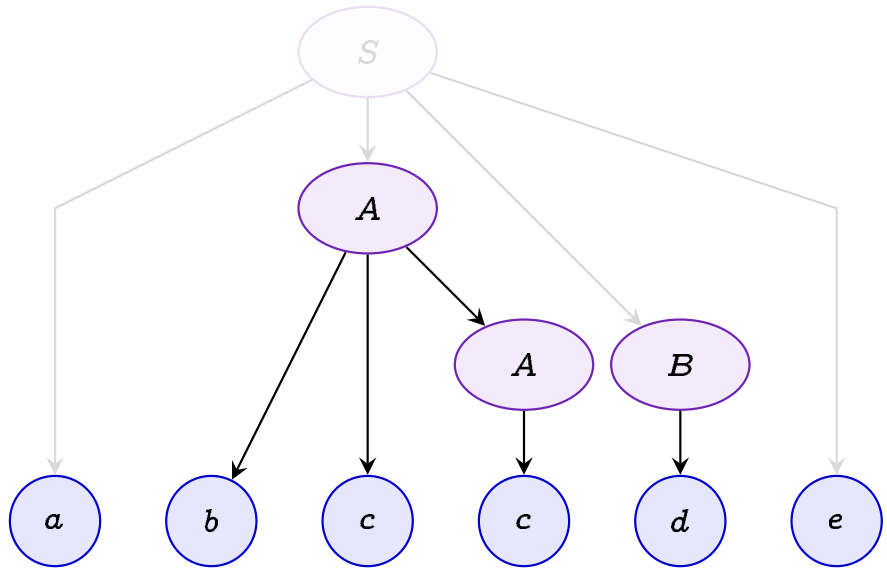
Näide:

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

Stack: $a\,A\,B\,e$

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

shift
shift
shift
shift
reduce $A \rightarrow c$
reduce $A \rightarrow b\,c\,A$
shift
reduce $B \rightarrow d$
shift
reduce $S \rightarrow a\,A\,B\,e$



$$
\begin{array}{lllll}
S & \Longrightarrow_{rm} & a\,A\,B\,e & \Longrightarrow_{rm} & a\,A\,d\,e \\
 & \Longrightarrow_{rm} & a\,b\,c\,A\,d\,e & \Longrightarrow_{rm} & a\,b\,c\,c\,d\,e
\end{array}
$$

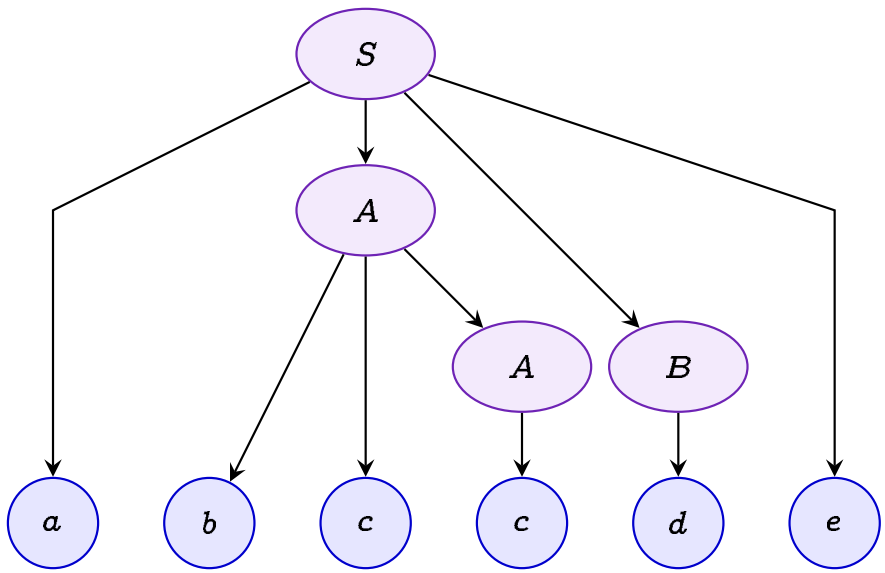
Shift-reduce parsimine

Näide:

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

Stack: S

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



shift
shift
shift
shift
reduce $A \rightarrow c$
reduce $A \rightarrow b\,c\,A$
shift
reduce $B \rightarrow d$
shift
reduce $S \rightarrow a\,A\,B\,e$

$$
\begin{array}{lllll}
S & \Longrightarrow_{rm} & a\,A\,B\,e & \Longrightarrow_{rm} & a\,A\,d\,e \\
 & \Longrightarrow_{rm} & a\,b\,c\,A\,d\,e & \Longrightarrow_{rm} & a\,b\,c\,c\,d\,e
\end{array}
$$

Shift-reduce parsimine

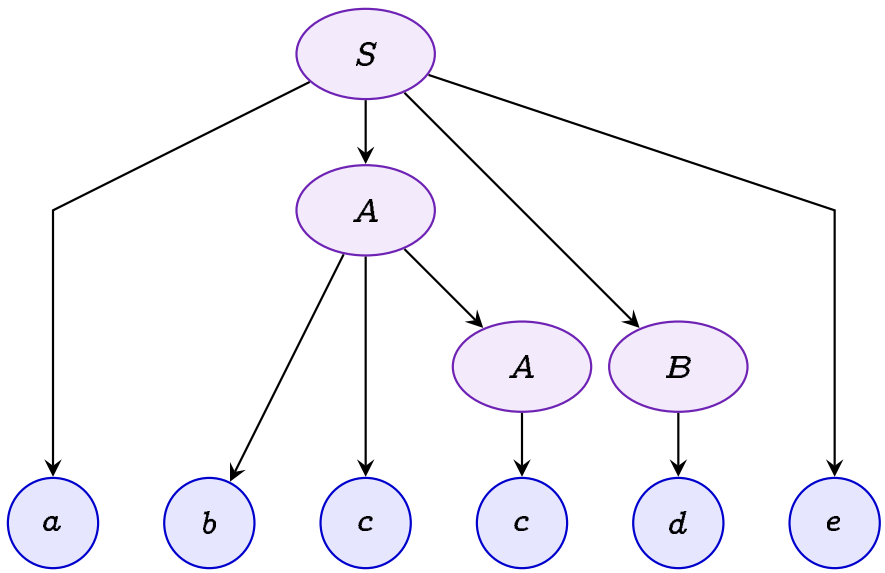
Näide:

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

Input String:

Stack:

shift
shift
shift
shift
reduce $A \rightarrow c$
reduce $A \rightarrow b\,c\,A$
shift
reduce $B \rightarrow d$
shift
reduce $S \rightarrow a\,A\,B\,e$



$$
\begin{array}{lllll}
S & \Longrightarrow_{rm} & a\,A\,B\,e & \Longrightarrow_{rm} & a\,A\,d\,e \\
 & \Longrightarrow_{rm} & a\,b\,c\,A\,d\,e & \Longrightarrow_{rm} & a\,b\,c\,c\,d\,e
\end{array}
$$

Shift-reduce parsimine

- Lausevormi nimetatakse paremlausevormiks (right-sentential form), kui ta on tuletatav paremderivatsiooni abil.
- Paremlausevormi γ pide (handle) on produktsiooni-reegel $A \to \beta$ ja sõnas γ alamssõna β esinemise positsioon, mis paremderivatsiooni eelmises lausevormis on asendatud mitteterminaliga A.
- Ekvivalentselt, paremlausevormi γ pide on alamstring β, selline et $S \Longrightarrow_{rm}^{\star} \delta A w \Longrightarrow_{rm} \delta \beta w = \gamma$, kus $\beta, \gamma, \delta \in V^{\star}$ ja $w \in T^{\star}$.
- 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

$$\begin{array}{lcl} S & \to & a\,A\,B\,e \\ A & \to & b\,c\,A \mid c \\ B & \to & d \end{array}$$

ja paremderivatsioon

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

Paremlausevormi $abcAde$ pide on bcA.

Shift-reduce parsimine

- Shift-reduce parseris tekib pide enne taandamist magasini tippu.
- Paremlausevormi prefikseid, mis ei lõpe kaugemal paremal kui selles lausevormis olev pide, nimetatakse elujõulisteks (viable).
- Elujõulised prefiksid on shift-reduce parseri võimalikeks magasinideks!
- NB! "Elujõuliste prefiksite 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 produktsioonireegel, mille parempoolde on kuhugi lisatud punkt.
- Näide: olgu antud grammatika

$$\begin{array}{lcl} S & \to & a\,A\,c \\ A & \to & A\,b \mid \varepsilon \end{array}$$

Tema LR(0)-elemendid on:

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

Shift-reduce parsimine

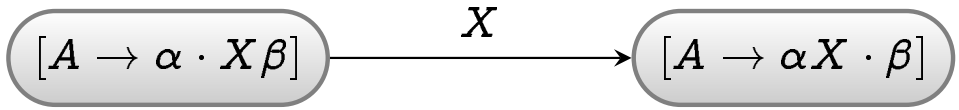
- Element $[A \to \alpha \cdot \beta]$ on elujõulise prefiksi φ suhtes kehtiv (valid), kui leidub paremderivatsioon

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

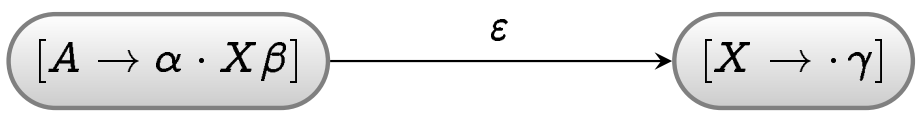
  ja $\delta\alpha = \varphi$.
- Kehtiv element $[A \to \alpha \cdot \beta]$ tähendab, et seni nähtud sisend on kooskõlas reegli $A \to \alpha\beta$ rakendamisega ja parser on juba ära tundnud α.
- Kehtiv element $[A \to \alpha\beta\cdot]$ tähendab, et parser on ära tundnud $\alpha\beta$ ja on kooskõlas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

Shift-reduce parsimine

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



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



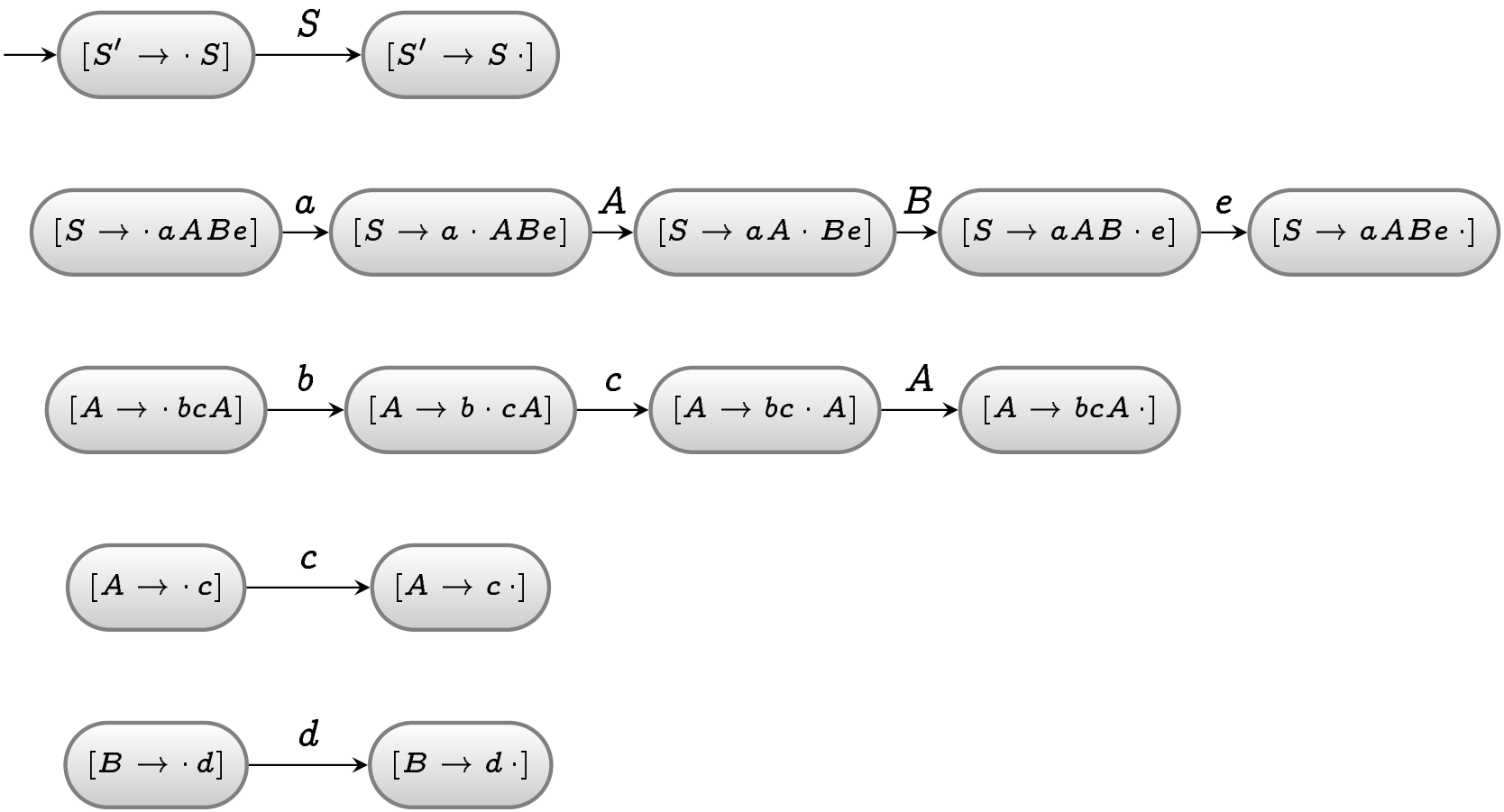
- Grammatika on laiendatud uue algsümboliga S' ja ühe uue reegliga $S' \to S$.
- Elementi $[S' \to \cdot S]$ sisaldav olek on automaadi algolekuks.

Shift-reduce parsimine

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

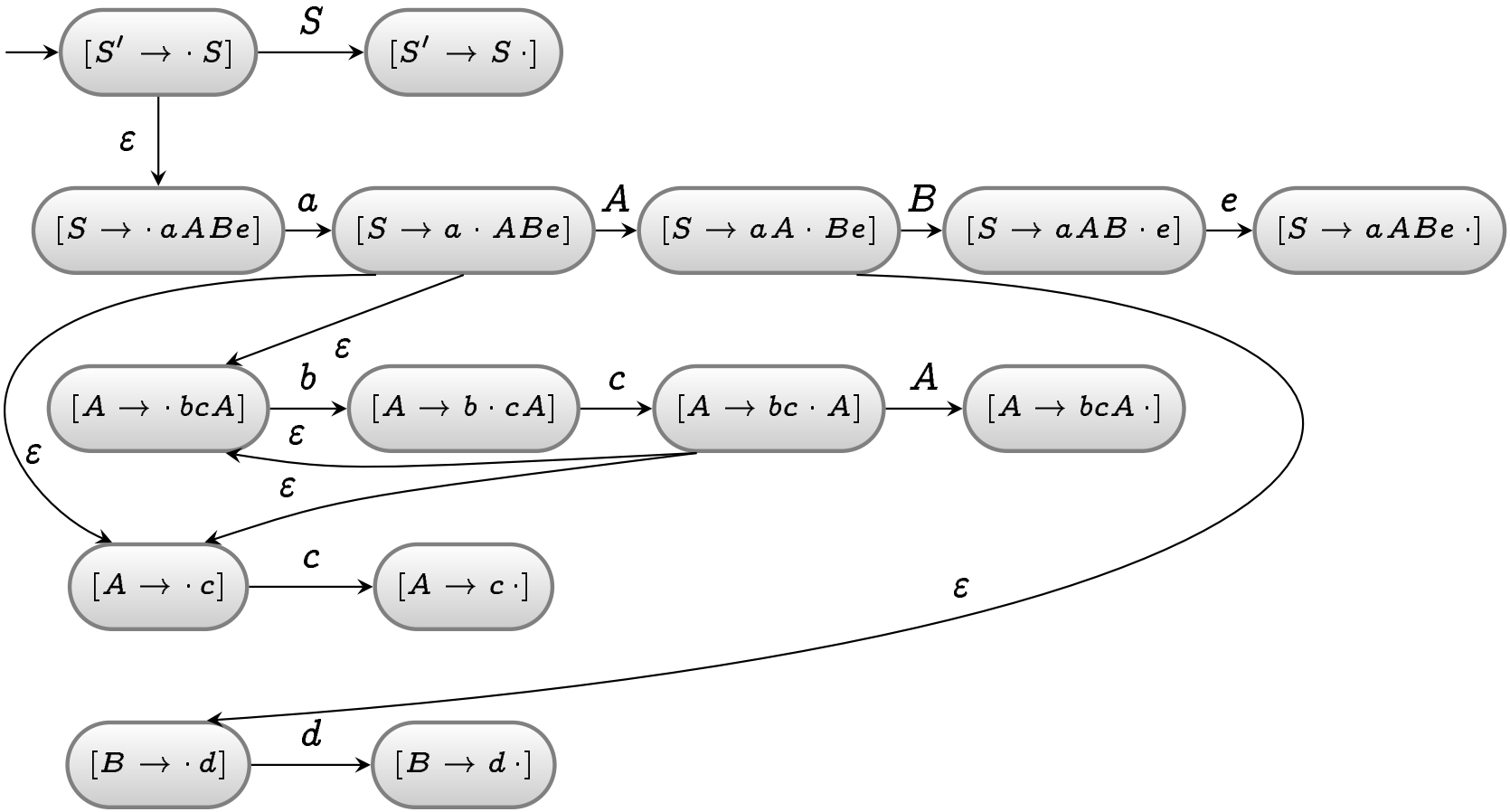
Shift-reduce parsimine

$$
\begin{array}{lcl}
S & \to & a\,A\,B\,e \\
A & \to & b\,c\,A \mid c \\
B & \to & d
\end{array}
$$



Shift-reduce parsimine

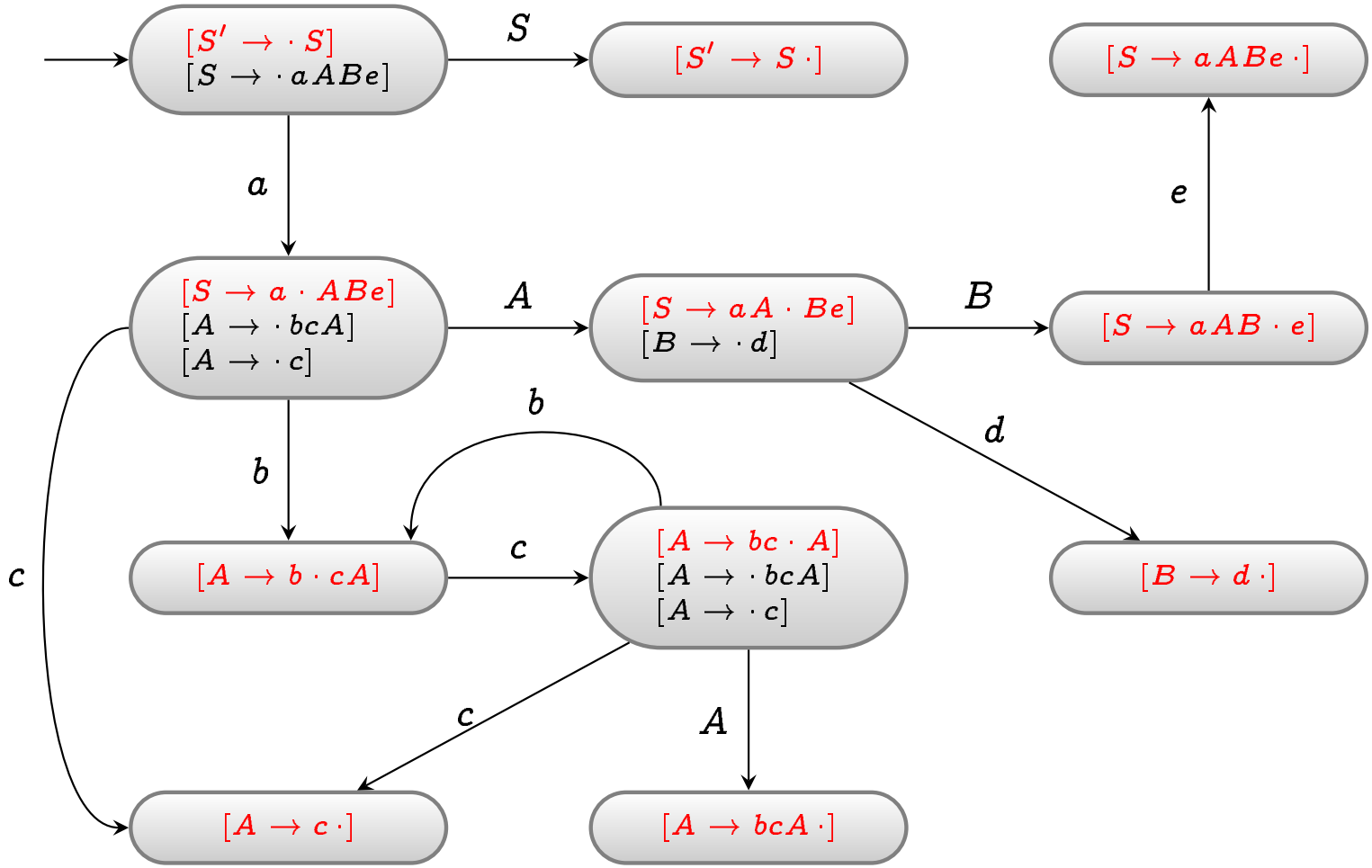
$$
\begin{array}{lcl}
S & \to & a\,A\,B\,e \\
A & \to & b\,c\,A \mid c \\
B & \to & d
\end{array}
$$



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 sulund-elementideks (closure items).
- Tuumelemendid identifitseerivad üheselt oleku.

Shift-reduce parsimine



Shift-reduce parsimine

LR(0) parsimisalgoritm:

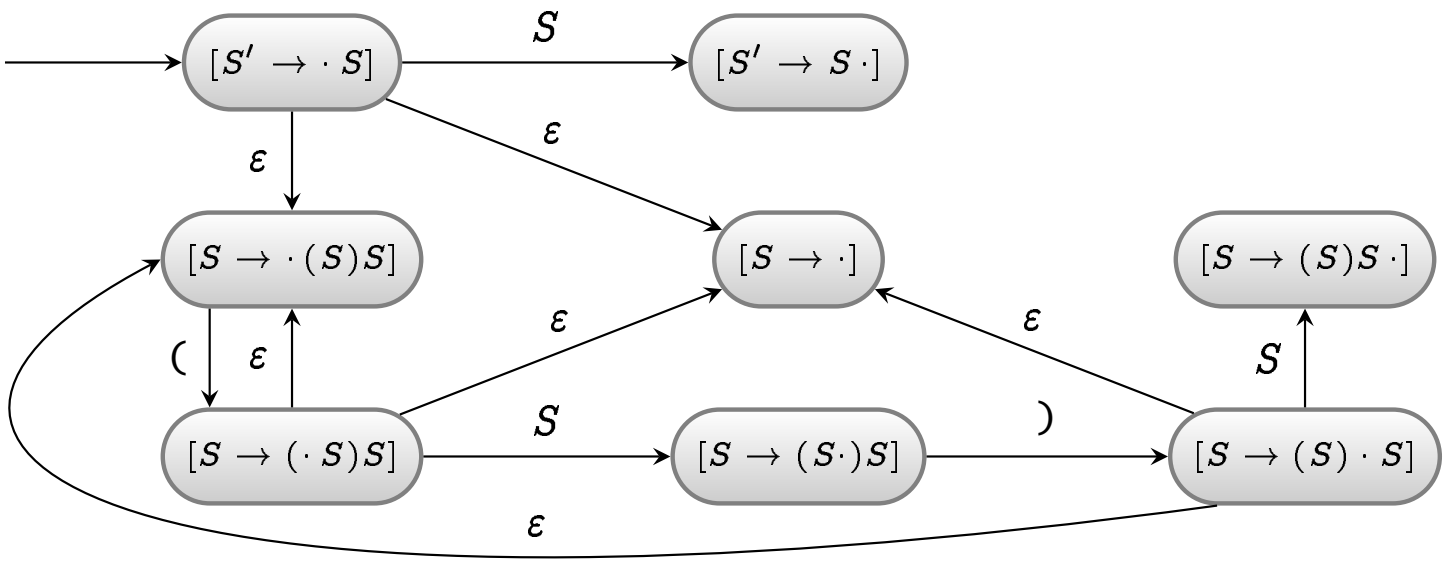
$push([S' \to \cdot\, S])$;
while true do {
  $state := top()$;
  if $\exists [A \to \alpha \cdot X\beta] \in state \ \wedge X \in T$ **then**
    $Y := getToken()$;
    if $\exists [A \to \alpha \cdot Y\beta] \in state$ **then** /* **shift** */
      $push([A \to \alpha Y \cdot \beta])$;
    else error;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

Shift-reduce parsimine

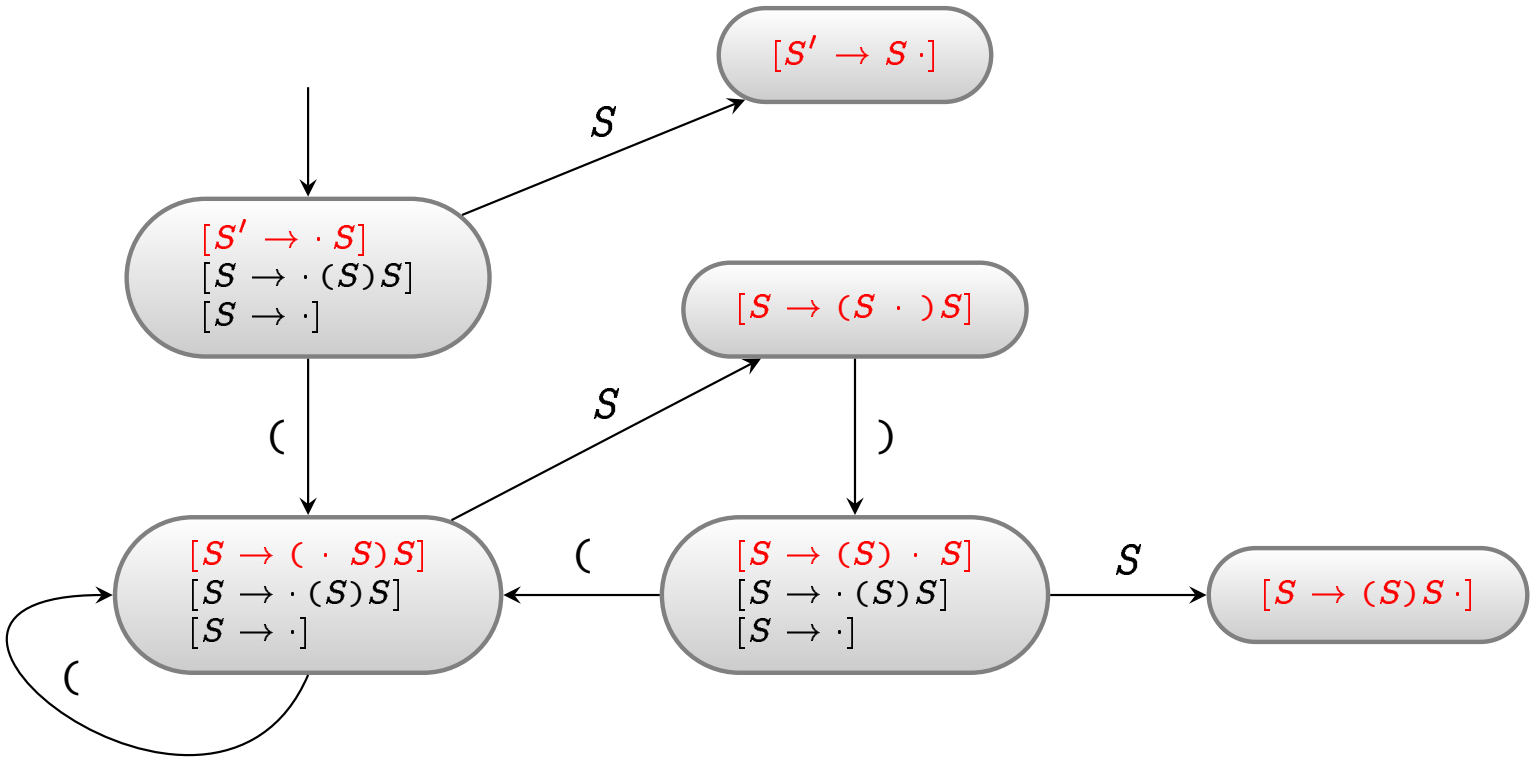
- Grammatika on LR(0), kui iga täiselement (so. element kujul $[A \to \alpha\cdot] \in \mathit{state}$) on teda sisaldava oleku state ainsaks elemendiks.
  - Shift-reduce konflikt: kui $\exists [B \to \alpha \cdot \beta] \in \mathit{state}$;
  - Reduce-reduce konflikt: kui $\exists [B \to \beta\cdot]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

Shift-reduce parsimine

$$
\begin{array}{lcl}
S' & \to & S \\
S & \to & (\; S \;) \; S \; \mid \; \varepsilon
\end{array}
$$



Shift-reduce parsimine



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 parsimine

SLR(1) parsimisalgoritm:

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

Shift-reduce parsimine

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

- Shift-reduce konflikt:

$$\forall [A \to \alpha \cdot X\beta] \in state \wedge X \in T \\ \Rightarrow \quad \neg(\exists [B \to \gamma \cdot] \in state \wedge X \in Follow(B))$$

- Reduce-reduce konflikt:

$$\forall [A \to \alpha \cdot] \in state \wedge [B \to \beta \cdot] \in state \\ \Rightarrow \quad Follow(A) \cap Follow(B) = \emptyset$$