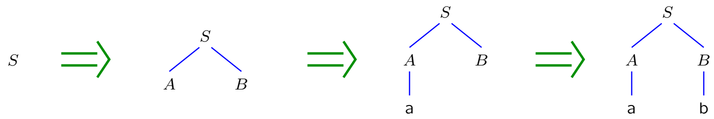
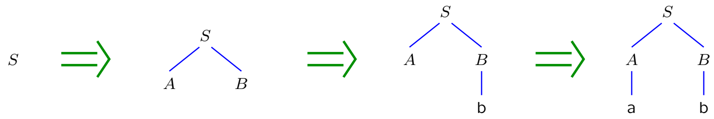
Vasen johto $S \Rightarrow AB \Rightarrow aB \Rightarrow ab$ esittää jäsennyspuun kasvattamista vasemmalta alkaen:



Samaan jäsennyspuuhun päästään myös johdolla $S \Rightarrow AB \Rightarrow Ab \Rightarrow ab$:



Yhteen jäsennyspuuhun liittyy aina tasan yksi vasen johto (sekä yleensä muita, ei-vasempia johtoja).

Kielioppi on *moniselitteinen* (ambiguous), jos siinä jollain lauseella on useampi kuin yksi vasemmanpuoleinen johto. Muuten kielioppi on *yksiselitteinen*. Toisin sanottuna kielioppi on moniselitteinen, jos ja vain jos siinä jollain lauseella on ainakin kaksi jäsennyspuuta.

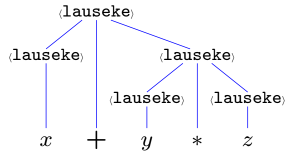
Tähänastiset esimerkkimme ovat kaikki olleet yksiselitteisiä.

Esimerkki 2.9: Muuttujien x, y ja z aritmeettiset lausekkeet voidaan kuvata kieliopilla

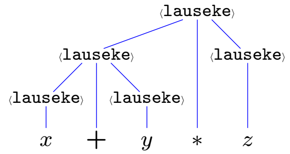
$$\langle \texttt{lauseke} \rangle \to \langle \texttt{lauseke} \rangle + \langle \texttt{lauseke} \rangle \mid \langle \texttt{lauseke} \rangle * \langle \texttt{lauseke} \rangle \mid (\ \langle \texttt{lauseke} \rangle\ ) \mid x \mid y \mid z.$$

Kielioppi on moniselitteinen, sillä lauseella $x + y * z$ on kaksi vasenta johtoa ja jäsennyspuuta:

$$\begin{aligned}
\langle \texttt{lauseke} \rangle &\Rightarrow \langle \texttt{lauseke} \rangle + \langle \texttt{lauseke} \rangle \\
&\Rightarrow x + \langle \texttt{lauseke} \rangle \\
&\Rightarrow x + \langle \texttt{lauseke} \rangle * \langle \texttt{lauseke} \rangle \\
&\Rightarrow x + y * \langle \texttt{lauseke} \rangle \\
&\Rightarrow x + y * z
\end{aligned}$$



$$\begin{aligned}
\langle \texttt{lauseke} \rangle &\Rightarrow \langle \texttt{lauseke} \rangle * \langle \texttt{lauseke} \rangle \\
&\Rightarrow \langle \texttt{lauseke} \rangle + \langle \texttt{lauseke} \rangle * \langle \texttt{lauseke} \rangle \\
&\Rightarrow x + \langle \texttt{lauseke} \rangle * \langle \texttt{lauseke} \rangle \\
&\Rightarrow x + y * \langle \texttt{lauseke} \rangle \\
&\Rightarrow x + y * z.
\end{aligned}$$

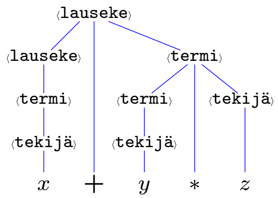


Sama kieli voidaan tuottaa seuraavalla yksiselitteisellä kieliopilla

$$\begin{aligned}
\langle \texttt{lauseke} \rangle &\to \langle \texttt{lauseke} \rangle + \langle \texttt{termi} \rangle \mid \langle \texttt{termi} \rangle \\
\langle \texttt{termi} \rangle &\to \langle \texttt{termi} \rangle * \langle \texttt{tekijä} \rangle \mid \langle \texttt{tekijä} \rangle \\
\langle \texttt{tekijä} \rangle &\to (\ \langle \texttt{lauseke} \rangle\ ) \mid x \mid y \mid z.
\end{aligned}$$

Lauseen $x + y * z$ yksikäsitteiset vasen johto ja jäsennyspuu ovat

$$\begin{aligned}
\langle \texttt{lauseke} \rangle &\Rightarrow \langle \texttt{lauseke} \rangle + \langle \texttt{termi} \rangle \\
&\Rightarrow \langle \texttt{termi} \rangle + \langle \texttt{termi} \rangle \\
&\Rightarrow \langle \texttt{tekijä} \rangle + \langle \texttt{termi} \rangle \\
&\Rightarrow x + \langle \texttt{termi} \rangle \\
&\Rightarrow x + \langle \texttt{termi} \rangle * \langle \texttt{tekijä} \rangle \\
&\Rightarrow x + \langle \texttt{tekijä} \rangle * \langle \texttt{tekijä} \rangle \\
&\Rightarrow x + y * \langle \texttt{tekijä} \rangle \\
&\Rightarrow x + y * z
\end{aligned}$$



Aritmeettisen lausekkeen jäsennyspuun avulla voidaan helposti laskea lausekkeen arvo, kun muuttujien arvot tunnetaan. Yleisemmin kääntäjä voi jäsennyspuun avulla *generoida koodia*, joka evaluoi lausekkeen. Edellisen sivun kielioppi noudattaa koulusta tuttua presedenssisääntöä, jonka mukaan kertolaskut lasketaan ennen yhteenlaskuja. Sen sijaan aiempi moniselitteinen kielioppi voi johtaa myös väärään laskujärjestykseen.

Todetaan vielä, että joillekin yhteydettömille kielille **ei ole olemassa** yksiselitteistä kielioppia. Esimerkki tällaisesta *luonnostaan moniselitteisestä* (inherently ambiguous) kielestä on $\{\, \mathrm{a}^i \mathrm{b}^j \mathrm{c}^k \mid i = j \text{ tai } j = k \,\}$ (todistus sivuutetaan).

Chomskyn normaalimuoto [Sipser s. 108–111]

Kielioppeja algoritmisesti käsiteltäessä on hyvä, jos ne ovat "siistejä".

Esimerkki 2.10: Tarkastellaan kielioppia

$$
\begin{array}{rcl}
S & \to & AB \\
A & \to & \mathrm{a} \\
B & \to & CC \\
C & \to & DD \\
D & \to & EE \\
E & \to & FF \\
F & \to & \varepsilon
\end{array}
$$

Kieliopin tuottama kieli on yksinkertaisesti $\{\,\mathrm{a}\,\}$. Kaikki johdot ovat kuitenkin varsin pitkiä, mikä on erilaisten algoritmien kannalta ongelmallista. □

Esitämme seuraavaksi, miten yhteydettömälle kieliopille voidaan löytää *Chomskyn normaalimuoto*, jossa erityisesti merkkijonon w johdon pituus on aina $2|w| - 1$.

Chomskyn normaalimuoto tarjoaa hyviä esimerkkejä yhteydettömien kielioppien yksinkertaistamisessa käytettävistä päättelyistä ja algoritmeista.

Määritelmä 2.11: Yhteydetön kielioppi (V, Σ, R, S) on *Chomskyn normaalimuodossa*, jos joukon R jokainen sääntö on jotain seuraavista muodoista:

1. $A \to BC$, missä $A, B, C \in V$ ja $B \neq S$ ja $C \neq S$,
2. $A \to a$, missä $A \in V$ ja $a \in \Sigma$, tai
3. $S \to \varepsilon$.

Normaalimuodosta seuraa erityisesti, että jos $S \stackrel{*}{\Rightarrow} w$ ja $|w| = n > 0$, niin johdon pituus on tasan $2n - 1$.

Lause 2.12: [Sipser Thm. 2.9] Mikä tahansa yhteydetön kieli voidaan tuottaa Chomskyn normaalimuodossa olevalla yhteydettömällä kieliopilla.

Esitämme, miten kielioppi muunnetaan Chomskyn normaalimuotoon kolmessa vaiheessa:

1. lisätään uusi lähtösymboli S_0 ja poistetaan säännöt $A \to \varepsilon$, missä $A \neq S_0$
2. poistetaan säännöt $A \to B$, missä $A, B \in V$
3. pilkotaan liian pitkät säännöt paloiksi.

Lemma 2.13: Mikä tahansa yhteydetön kieli voidaan tuottaa yhteydettömällä kieliopilla, jossa

1. lähtösymboli ei esiinny minkään säännön oikealla puolella ja
2. kieliopissa ei ole sääntöä $A \to \varepsilon$ millään A, joka *ei ole* lähtösymboli.

Todistus: Olkoon $G = (V, \Sigma, R, S)$ yhteydetön kielioppi. Muodostamme ehdot toteuttavan kieliopin $G' = (V', \Sigma, R', S_0)$, jolla $L(G') = L(G)$.

Otetaan käyttöön uusi lähtösymboli $S_0 \notin V$ ja asetetaan $V' = V \cup \{\,S_0\,\}$. Laitetaan joukkoon R' sääntö $S_0 \to S$. Jos $\varepsilon \in L(G)$, lisätään myös sääntö $S_0 \to \varepsilon$. Muuttujalle S_0 ei tule muita sääntöjä, eikä se tule minkään säännön oikealle puolelle.

Määritellään nyt *nollautuvien* muuttujien joukko

$$\mathrm{Null} = \left\{\, A \in V \mid A \stackrel{*}{\Rightarrow} \varepsilon \,\right\}.$$

Joukko Null voidaan helposti laskea iteratiivisella algoritmilla.

```
EtsiNollautuvat(V, Σ, R, S)
    Null := ∅
    VanhaNull := Null
    for A ∈ V
        do if joukossa R on sääntö A → ε
            then Null := Null ∪ { A }
    while Null ≠ VanhaNull          ▷ toista kunnes Null ei muutu
        do VanhaNull := Null
            for A ∈ V
                do if joukossa R on sääntö A → B_1 ... B_k
                      missä B_i ∈ Null kaikilla i
                    then Null := Null ∪ { A }
    return Null
```

Perusidea on, että jos $B \in$ Null, niin sääntö $A \to uBv$ korvataan säännöillä

$$A \to uv \mid uBv.$$

Tällöin jos alkuperäisessä kieliopissa on ε-sääntöä käyttävä johto

$$A \Rightarrow uBv \Rightarrow uv$$

niin uudessa kieliopissa voidaan jättää uBv väliin ja johtaa suoraan

$$A \Rightarrow uv.$$

Vastaavasti jos $B, C \in$ Null, niin sääntö $A \to uBvCw$ korvataan säännöillä

$$A \to uvw \mid uvCw \mid uBvw \mid uBvCw,$$

jne. Tämän jälkeen ε-säännöt ovat jääneet turhiksi ja voidaan poistaa.

Käydään siis läpi kaikki joukon R säännöt $A \to w$. Kirjoitetaan w muotoon

$$w = u_0 A_1 u_1 A_2 u_2 \ldots A_k u_k,$$

missä kukin A_i on nollautuva muuttuja ja toisaalta mikään u_i ei sisällä nollautuvia muuttujia. Jos w ei sisällä nollautuvia muuttujia, niin $k = 0$. Lisätään nyt joukkoon R' kaikki 2^k sääntöä

$$A \to u_0 x_1 u_1 x_2 u_2 \ldots x_k u_k,$$

missä x_i on joko A_i tai ε, *paitsi* ei sääntöä $A \to \varepsilon$ mikäli se esiintyy.

On helppo nähdä, että saatu kielioppi toteuttaa vaaditut ehdot. □

Esimerkki 2.14: Sovelletaan konstruktiota kielioppiin

$$\begin{array}{rcl} S & \to & BSB \mid A \\ A & \to & \mathrm{a}A \mid \mathrm{aa} \\ B & \to & \mathrm{b}B \mid \varepsilon. \end{array}$$

Nyt Null $= \{B\}$. Siis säänntö $S \to BSB$ korvataan säännöillä

$$S \to S \mid SB \mid BS \mid BSB,$$

ja sääntö $B \to \mathrm{b}B$ säännöillä

$$B \to \mathrm{b} \mid \mathrm{b}B.$$

Muut säännöt paitsi lukuunottamatta ε-sääntöä jäävät ennalleen. Lisätään vielä alkusymboli ja sääntö $S_0 \to S$, muttei sääntöä $S_0 \to \varepsilon$, koska $S \overset{*}{\not\Rightarrow} \varepsilon$. Saadaan kielioppi

$$\begin{array}{rcl} S_0 & \to & S \\ S & \to & S \mid SB \mid BS \mid BSB \mid A \\ A & \to & \mathrm{a}A \mid \mathrm{aa} \\ B & \to & \mathrm{b} \mid \mathrm{b}B. \end{array}$$

□

Seuraava vaihe on poistaa *yksikkösäännöt* eli säännöt muotoa $A \to B$, missä $A, B \in V$.

Lemma 2.15: Mikä tahansä yhteydetön kieli voidaan tuottaa yhteydettömällä kieliopilla, jossa ei ole yksikkösääntöjä.

Todistus: Kaikilla $A \in V$ olkoon Unit(A) niiden muuttujien joukko, jotka voidaan johtaa A:sta pelkillä yksikkösäännöillä, A itse mukaanlukien. Joukot Unit(A) on helppo laskea iteratiivisesti.

Kieliopin muuttujat ja päätesymbolit pidetään ennallaan, samoin lähtösymboli. Sääntöihin laitetaan $A \to w$ kaikilla $A \in V$ ja $w \in (V \cup \Sigma)^*$, joilla

1. $B \in$ Unit(A),
2. $B \to w$ on sääntö alkuperäisessä kieliopissa ja
3. w ei ole muuttujasymboli.

Ehtojen 1 ja 2 perusteella on selvää, että uusien sääntöjen joukkoon ei tule mitään, mitä siellä ei saisi olla. Jos $B \in \text{Unit}(A)$ ja $B \Rightarrow w$, niin $A \overset{*}{\Rightarrow} w$, joten $A \to w$ voidaan lisätä.

Ainoa ongelma näyttäisi olevan, että ehdon 3 nojalla pudotetaan pois muotoa $B \to C$ olevat säännöt. Kuitenkin jos $B \in \text{Unit}(A)$ ja $B \to C$ on sääntö, niin $C \in \text{Unit}(A)$, joten vuorostaan muotoa $C \to w$ olevat säännöt tuovat mukaan uudet säännöt $A \to w$. Siis uudet säännöt tuottavat samat merkkijonot kuin vanhat. □

Esimerkki 2.16: Jatketaan edellisen esimerkin lopputuloksesta

$$\begin{array}{rcl} S_0 & \to & S \\ S & \to & S \mid SB \mid BS \mid BSB \mid A \\ A & \to & \mathsf{a}A \mid \mathsf{aa} \\ B & \to & \mathsf{b} \mid \mathsf{b}B. \end{array}$$

Nyt $\text{Unit}(S_0) = \{ S_0, S, A \}$, $\text{Unit}(S) = \{ S, A \}$, ja $\text{Unit}(X) = \{ X \}$ muuten. Siis muuttujalle S_0 tulee omien alkuperäisten lisäksi kaikki muuttujien S ja A ei-yksikkösäännöt:

$$S_0 \to SB \mid BS \mid BSB \mid \mathsf{a}A \mid \mathsf{aa}.$$

Samoin S:lle lisätään A:n säännöt:

$$S \to SB \mid BS \mid BSB \mid \mathsf{a}A \mid \mathsf{aa}.$$

Ottamalla vielä kaikki vanhat ei-yksikkösäännöt saadaan kielioppi

$$\begin{array}{rcl} S_0 & \to & SB \mid BS \mid BSB \mid \mathsf{a}A \mid \mathsf{aa} \\ S & \to & SB \mid BS \mid BSB \mid \mathsf{a}A \mid \mathsf{aa} \\ A & \to & \mathsf{a}A \mid \mathsf{aa} \\ B & \to & \mathsf{b} \mid \mathsf{b}B. \end{array}$$

□

Lauseen 2.12 todistus: Olkoon $G = (V, \Sigma, R, S)$ yhteydetön kielioppi. Muodostamme Chomskyn normaalimuodossa olevan kieliopin, joka tuottaa saman kielen. Lemmojen 2.13 ja 2.15 nojalla voimme olettaa kieliopista G, että

1. S ei esiinny minkään säännön oikealla puolella,
2. $A \to \varepsilon \notin R$ kaikilla $A \in V - \{ S \}$, ja
3. yksikkösääntöjä ei ole.

Siis sääntö $A \to u_1 \dots u_k$, missä $u_i \in V \cup \Sigma$, voi rikkoa normaalimuotoa kahdella tavalla:

1. joko $k = 2$ ja jokin u_i on päätesymboli tai
2. $k \geq 3$.

Otetaan jokaista päätesymbolia $a \in \Sigma$ kohti käyttöön uusi muuttuja X_a ja lisätään sääntö $X_a \to a$. Kaikissa normaalimuotoa rikkovissa säännöissä, joissa esiintyy a, korvataan se X_a:lla.

Sääntö $A \to u_1 \dots u_k$, missä $k \geq 3$, korvataan $k - 1$ uudella säännöllä

$$\begin{array}{rcl} A & \to & u_1 A_2 \\ A_2 & \to & u_2 A_3 \\ & \dots & \\ A_{k-2} & \to & u_{k-2} A_{k-1} \\ A_{k-1} & \to & u_{k-1} u_k \end{array}$$

missä $A_2, \dots, A_{k-1}$ ovat uusia muuttujia. □

Esimerkki 2.17: Jatketaan edellistä esimerkkiä

$$\begin{aligned}
S_0 &\to SB \mid BS \mid BSB \mid \mathsf{a}A \mid \mathsf{aa}\\
S &\to SB \mid BS \mid BSB \mid \mathsf{a}A \mid \mathsf{aa}\\
A &\to \mathsf{a}A \mid \mathsf{aa}\\
B &\to \mathsf{b} \mid \mathsf{b}B.
\end{aligned}$$

Kielioppi tulee muotoon

$$\begin{aligned}
S_0 &\to SB \mid BS \mid BA_1 \mid X_\mathsf{a}A \mid X_\mathsf{a}X_\mathsf{a}\\
A_1 &\to SB\\
S &\to SB \mid BS \mid BA_2 \mid X_\mathsf{a}A \mid X_\mathsf{a}X_\mathsf{a}\\
A_2 &\to SB\\
A &\to X_\mathsf{a}A \mid X_\mathsf{a}X_\mathsf{a}\\
B &\to \mathsf{b} \mid X_\mathsf{b}B\\
X_\mathsf{a} &\to \mathsf{a}\\
X_\mathsf{b} &\to \mathsf{b}.
\end{aligned}$$

□

Yhteydettömän kieliopin jäsennysongelma

Yhteydettömän kieliopin jäsennysongelmalla tarkoitetaan laskentaongelmaa

Annettu: yhteydetön kielioppi G, merkkijono w
Kysymys: päteekö $w \in L(G)$.

Myönteisessä tapauksessa usein halutaan myös w:n johto tai jäsennyspuu.

Ongelma voidaan periaatteessa ratkaista seuraavasti:

1. Muodosta Chomskyn normaalimuodossa oleva G', jolla $L(G') = L(G)$.
2. Jos $w = \varepsilon$ ja G' sisältää säännön $S \to \varepsilon$, vastaa *kyllä*.
3. Muuten luettele kaikki pituudeltaan $2|w| - 1$ olevat kieliopin G' johdot. Jos jokin näistä on johto merkkijonolle w, vastaa *kyllä*; muuten vastaa *ei*.

Ratkaisu perustuu havaintoon, että Chomskyn normaalimuodossa w:n johdon pituus on aina $2|w| - 1$, kun $w \neq \varepsilon$. Koska pituutta $2|w| - 1$ olevia johtoja voi olla paljon, menetelmä on käytännössä liian tehoton.

Tehokkaampi vaihtoehto on CYK-algoritmi (Cocke, Younger, Kasami), joka ratkaisee ongelman ajassa $O(|w|^3)$. Myönteisessä tapauksessa algoritmi antaa myös jäsennyspuun. Käytännössä myös $O(|w|^3)$ on kuitenkin usein liian hidasta.

Yksi tärkeimmistä jäsennysongelman sovelluksista on ohjelman jäsennys ohjelmointikielen kääntäjässä. Niissä käytetään tehokkaampia menetelmiä, jotka edellyttävät, että kielioppi on jossain rajoitetussa muodossa (tärkeimmät LL(k) ja LR(k)). Emme tällä kurssilla puutu näihin menetelmiin.

Esitämme tässä CYK-algoritmin lyhyesti (huom. kurssikirjassa sivulla 267). Voimme olettaa, että kielioppi $G = (V, \Sigma, R, S)$ on jo muunnettu Chomskyn normaalimuotoon.

Tarkastellaan jotakin johtoa $A \overset{*}{\Rightarrow} w$, missä $w = w_1 \dots w_n \in \Sigma^n$ ja $n \geq 2$. Koska kielioppi on Chomskyn normaalimuodossa, ensimmäisen askeleen täytyy olla $A \Rightarrow BC$, missä $B \overset{*}{\Rightarrow} w_1 \dots w_i$ ja $C \overset{*}{\Rightarrow} w_{i+1} \dots w_n$ jollakin i.

Saadaan seuraava rekursiivinen proseduuri Derives(A, w), joka kaikilla muuttujilla $A \in V$ ja päätemerkkijonoilla $w \in \Sigma^*$ palauttaa True, jos $A \overset{*}{\Rightarrow} w$.

```
Derives(A, w)
    if w = ε and (A → ε) ∈ R
        then return True        ▷ tyhjä merkkijono
    elseif w = a ∈ Σ and (A → a) ∈ R
        then return True        ▷ yksi merkki
    else olkoon w = w1 ... wn, missä wi ∈ Σ ja n ≥ 2
            for kaikilla säännöillä (A → BC) ∈ R
                do for i ← 1 to n − 1
                        do if Derives(B, w1 ... wi) and Derives(C, wi+1 ... wn)
                            then return True
    return False
```

Nyt $w \in L(G)$, jos ja vain jos Derives(S, w) palauttaa True.