- Kontekstiton kielioppi G on *moniselitteinen*, jos jollakin G:n lauseella x on kaksi erilaista G:n mukaista jäsennyspuuta
- Muuten kielioppi on *yksiselitteinen*
- Kontekstiton kieli, jonka tuottavat kieliopit ovat kaikki moniselitteisiä, on *luonnostaan moniselitteinen*
- Esimerkkejä:
  - Kielioppi G'_{expr} on moniselitteinen
  - Kieliopit G_{expr} ja G_{match} yksiselitteisiä
  - Kieli $L_{\text{expr}} = L(G'_{\text{expr}})$ ei ole luonnostaan moniselitteinen, koska sillä on myös yksiselitteinen kielioppi G_{expr}
  - Kieli $\{a^i b^j c^k \mid i = j \text{ tai } j = k\}$ on luonnostaan moniselitteinen (todistus sivuutetaan)

4.5 Rekursiivisesti etenevä jäsentäminen

4.5.1 LL(1)-kielioppi

LL(1)-kielet ovat suppea mutta hyödyllinen kieliluokka. Niille voidaan helposti laatia rekursiivisesti etenevä jäsentäjä, joka lukee merkkijonoa vasemmalta oikealle ja joka asekeleella simuloi annetun kieliopin sääntöjä. Koska *LL(1)-kieliopissa* jokainen sääntö on yksikäsitteisesti määritelty, kun seuraava syötemerkki tunnetaan, voidaan merkkijonon vasemman johdon tuottava jäsennys suorittaa tietokoneohjelmana. (Nimitys LL(1)-kielioppi tuleekin sanoista "left to right scan, producing left parse with 1 symbol lookahead".)

- Tarkastellaan seuraavaa kielioppia G:

$$\begin{aligned} E &\rightarrow T + E \mid T - E \mid T \\ T &\rightarrow a \mid (E) \end{aligned}$$

- Muokataan G:stä välikkeen E produktiot "tekijöimällä" ekvivalentti kielioppi G':

$$\begin{aligned} E &\rightarrow TE' \\ E' &\rightarrow +E \mid -E \mid \epsilon \\ T &\rightarrow a \mid (E) \end{aligned}$$

- nyt G':n lauseille voidaan helposti muodostaa vasemmat johdot suoraan lähtösymbolista E alkaen: Jäsennyksen joka vaiheessa tavoitteena olevan lauseen *seuraava merkki* määrää yksikäsitteisesti sen, mikä produktio valitaan lavennettavana olevaan välikkeeseen

4.5.2 LL(1)-kieliopin jäsennysohjelma

- Jäsennysohjelma muodostuu välikkeitä vastaavista rekursiivisista funktioista
- Esim. G':n mukainen jäsentäjä pseudokoodina:

```
function E
begin
T;
E';
end

function E'
begin
if (next == '+') then
begin
    tee jotakin;
    next = getnext;
    E;
end
else
    begin
        if (next == '-') then
            begin
                tee jotakin;
                next = getnext;
                E;
            end
        else tee jotakin;
    end
end

function T
begin
if (next == 'a') then
```

```
    begin
        tee jotakin;
        next = getnext;
        return;
    end
else
    begin
        if (next == '(') then
            begin
                tee jotakin;
                next = getnext;
                E;
                if (next ≠ ')')
                        ERROR;
                next = getnext;
            end
        else ERROR;
    end
end
```

PÄÄOHJELMASSA:
next = *getnext*;
E;

Esim. syötejonon `a-(a+a)` jäsennys, kun käsky "tee jotakin' tulostaa produktiot:

```
E → TE'
T → a
E' → -E
E → TE'
T → (E)
E → TE'
T → a
E' → +E
E → TE'
T → a
E' → ϵ
E' → ϵ.
```

Tulostus vastaa vasenta johtoa:

$$\begin{aligned} E &\Rightarrow TE' \Rightarrow aE' \Rightarrow a - E \Rightarrow a - TE' \\ &\Rightarrow a - (E)E' \Rightarrow a - (TE')E' \\ &\Rightarrow a - (aE')E' \Rightarrow a - (a + E)E' \\ &\Rightarrow a - (a + TE')E' \Rightarrow a - (a + aE')E' \\ &\Rightarrow a - (a + a)E' \Rightarrow a - (a + a) \end{aligned}$$

4.5.3 LL(1)-kielioppien yleinen muoto

LL(1)-kieliopeissa sallitaan rajoitetussa määrin myös

- Produktioita, joiden oikeat puolet alkavat välikkeellä, sekä
- *Tyhjentyviä* välikkeitä A, joilla $A \Rightarrow^* \epsilon$

Esim. kielen $a^*b \cup c^*d$ tuottava kielioppi:

$$\begin{array}{lcll} S & \rightarrow & Ab & \mid \; Cd \\ A & \rightarrow & aA & \mid \; \epsilon \\ C & \rightarrow & cC & \mid \; \epsilon. \end{array}$$

Kielioppi on LL(1), vaikka ensimmäiseksi sovellettavaa produktiota ei voikaan päätellä pelkästään S:n produktioiden perusteella

4.5.4 Apukäsite: päätemerkkijoukot FIRST ja FOLLOW

Määritellään annetun kieliopin $G = (V, \Sigma, P, S)$ välikkeisiin liittyvät päätemerkkijoukot:

- FIRST(A) = A:sta johdettavien päätejonojen 1. merkit + ϵ, jos A tyjentyvä
- FOLLOW(A) = ne päätemerkit, jotka voivat seurata A:ta jossain G:n lausejohdoksessa + ϵ, jos A voi sijaita lausejohdoksen lopussa
- Formaalisti:

$$\begin{aligned}
\text{FIRST}(A) &= \{a \in \Sigma \mid A \Rightarrow^* ax \text{ jollakin } x \in \Sigma^*\} \\
&\quad \cup \{\epsilon \mid A \Rightarrow^* \epsilon\}; \\
\text{FOLLOW}(A) &= \{a \in \Sigma \mid S \Rightarrow^* \alpha A a \beta \text{ joillakin } \alpha, \beta \in V^*\} \\
&\quad \cup \{\epsilon \mid S \Rightarrow^* \alpha A \text{ jollakin } \alpha \in V^*\}.
\end{aligned}$$

FIRST-joukkojen määritelmä laajennetaan mielivaltaisille merkkijonoille, ja edelleen merkkijonojoukkoihin:

$$\text{FIRST}(\epsilon) = \{\epsilon\};$$

$$\text{FIRST}(a) = \{a\} \qquad \text{kaikilla } a \in \Sigma;$$

$$\text{FIRST}(X_1 \ldots X_k) =$$

$$\begin{cases}
\text{FIRST}(X_1) \cup \ldots \cup \text{FIRST}(X_i) - \{\epsilon\}, \\
\qquad \text{jos } \epsilon \in \text{FIRST}(X_1), \ldots, \text{FIRST}(X_{i-1}), \\
\qquad \epsilon \notin \text{FIRST}(X_i); \\
\text{FIRST}(X_1) \cup \ldots \cup \text{FIRST}(X_k), \\
\qquad \text{jos } \epsilon \in \text{FIRST}(X_i) \text{ kaikilla } i = 1, \ldots, k;
\end{cases}$$

Määritelmä: Kielioppi on LL(1)-muotoinen, jos sen millä tahansa kahdella samaan välikkeeseen A liittyvällä produktiolla $A \to \omega_1$ ja $A \to \omega_2$, $\omega_1 \neq \omega_2$, on voimassa:

$$\begin{aligned}
&\text{FIRST}(\{\omega_1\}\text{FOLLOW}(A)) \\
&\quad \cap \quad \text{FIRST}(\{\omega_2\}\text{FOLLOW}(A)) \quad = \quad \emptyset.
\end{aligned}$$

Esim.1 Onko seuraava kielioppi LL(1)-muodossa?

$S \to Ab|Cd$
$A \to aA|\epsilon$
$C \to cC|\epsilon$

Sääntöpari $S \to Ab$ ja $S \to Cd$:

$FIRST(\{Ab\}FOLLOW(S)) = FIRST(\{Ab\}\{\epsilon\})$, koska S voi esiintyä vain johdon ensimmäisenä lausejohdoksena (pelkkä aloitussymboli S, jota siis seuraa ϵ).

$FIRST(Ab\epsilon) = a, b$, koska A:ta lavennettaessa 1. merkki voi olla a tai ϵ, jolloin seuraava merkki (b) onkin merkkijonon Ab 1. merkki. Huom! ϵ ei voi olla merkkijonon $\{Ab\}\{\epsilon\}$ 1. merkki, sillä välissä on aina b, joka ei tyhjene mihinkään.

Samaan tapaan:
$FIRST(\{Cd\}FOLLOW(S)) = FIRST(\{Cd\}\{\epsilon\}) = \{c, d\}$

Sääntöparin $A \to aA$ ja $A \to \epsilon$ käsittely:

$FIRST(\{aA\}FOLLOW(A)) = FIRST(\{aA\}\{b\})$, koska A:ta voi seurata vain b jossain lausejohdoksessa - se ei voi olla merkkijonon viimeinen, koska S:stä lähtiessä sitä seuraa b!

$FIRST(\{aA\}\{b\}) = FIRST(\{aAb\}) = \{a\}$
$FIRST(\{\epsilon\}FOLLOW(A)) = FIRST(\{\epsilon b\}) = \{b\}$

Kolmas sääntöpari:

$FIRST(\{cC\}FOLLOW(C)) = FIRST(\{cC\}\{d\}) = \{c\}$
$FIRST(\{\epsilon\}FOLLOW(C)) = FIRST(\{\epsilon\}\{d\}) = \{d\}$

Eli mikään sääntöpari ei riko LL(1)-ehtoa (leikkaus tyhjä).

Esim. 2

$S \to (L)|a$
$L \to L, S|S$

Ensimmäinen sääntöpari:

$FIRST(\{(L)\}FOLLOW(S)) = FIRST(\{(L)\}\{')','\,,\epsilon\}) = \{'('\}$
$FIRST(\{a\}FOLLOW(S)) = \{a\}$
eli leikkaus tyhjä - ok.

Toinen sääntöpari:

$FIRST(\{L, S\}FOLLOW(L)) = FIRST(\{L, S\}\{')','\,'\}) = FIRST(L) = \{'('\,'\,a'\}$
$FIRST(\{S\}FOLLOW(L)) = FIRST(\{S\}\{')','\,'\}) = FIRST(S) = \{'('\,'\,a'\}$
eli leikkaus epätyhjä! Ei siis LL(1).

Huom! $FIRST(\{L, S\}\{')','\,'\}) = FIRST(L)$, koska L ei voi tyhjentyä (sitä ei voi korvata ϵ:illa missään johdoksessa - siispä merkkijonon 1. merkki on L:n 1. merkki. Samoin $FIRST(\{S\}\{')','\,'\}) = FIRST(S)$.

4.5.5 *LL(1)-kieliopin rekursiivisesti etenevän jäsentäjän muodostaminen yleisesti:

Apurutiinit:

token *getnext*;
    Seuraavan päätesymbolin selaus
... — voi olla > 1 kirjainmerkkiä.*
void ERROR(char* *msg*);
    Virheiden käsittely; parametri *msg*
... sisältää virheilmoitustekstin.*

Välikettä A vastaava funktio:

function A /*A:n produktiot $A \to \omega_1 \mid \ldots \mid \omega_n$*/
begin
    if (*next* **in** $[a_{11}, \ldots, a_{1m_1}]$) /*FIRST($\{\omega_1\}$FOLLOW($A$)) = $\{a_{11}, \ldots, a_{1m_1}\}$*/
    begin /*produktio $A \to \omega_1$*/
        $parse(\omega_1)$;
    end
    else
    ⋮
    if (*next* **in** $[a_{n1}, \ldots, a_{nm_n}]$) /*FIRST($\{\omega_n\}$FOLLOW($A$)) = $\{a_{n1}, \ldots, a_{nm_n}\}$*/
    begin /*produktio $A \to \omega_n$*/
        $parse(\omega_n)$;
    end
    else
        ERROR("A `cannot start with this.`")
end

Tässä lyhennemerkintä "$parse(\omega_i)$" tarkoittaa seuraavalla tavalla muodostettavaa käskyjonoa:
$parse(X_1 \ldots X_k) \equiv parse(X_1); \ldots ; parse(X_k)$,

missä

$parse(a) \equiv$
if $next \neq a$ **then** ERROR('a `expected.`');

next := *getnext*; (a on päätemerkki)

$parse(B) \equiv B$; (B on välike)

4.5.6 Kielioppien muokkaaminen LL(1)-muotoon

- LL(1)-kieliopit ovat varsin suppea kielioppiluokka
- Seuraavilla muunnoksilla voidaan joitakin "melkein' LL(1)-muotoisia kielioppeja muuntaa tähän muotoon:

1. VASEN TEKIJÖINTI

- Kielioppi, jossa on produktiot

$$A \to \alpha\beta_1 \mid \alpha\beta_2, \qquad \alpha \neq \epsilon,\ \beta_1 \neq \beta_2$$

  ei voi olla LL(1)-muotoinen

- Parannus: otetaan käyttöön uusi välike A' ja korvataan em. produktiot produktioilla:

$$\begin{aligned} A &\to \alpha A' \\ A' &\to \beta_1 \mid \beta_2, \end{aligned}$$

  missä α on $\alpha\beta_1$:n ja $\alpha\beta_2$:n pisin yhteinen alkuosa

2. VÄLITTÖMÄN VASEMMAN REKURSION POISTAMINEN

- Kielioppi on *vasemmalle rekursiivinen*, jos jollakin välikkeellä A ja merkkijonolla γ on

$$A \Rightarrow^+ A\gamma,$$

  missä merkintä $\alpha \Rightarrow^+ \beta$ tarkoittaa, että α:sta voidaan johtaa β johdolla, jonka pituus on vähintään yksi askel

- Vasemmalle rekursiivinen kielioppi ei voi täyttää LL(1)-ehtoa

- *Välitön* vasen rekursio, so. suorat johdot $A \Rightarrow A\gamma$, voidaan välttää korvaamalla produktiot

$$A \to A\beta \mid \alpha, \qquad \beta \neq \epsilon,$$

produktioilla

$$\begin{aligned} A &\to \alpha A' \\ A' &\to \beta A' \mid \epsilon \end{aligned}$$

- Muotoa $A \to A$ olevat produktiot voidaan yksinkertaisesti jättää pois

Jokainen kontekstiton kielioppi voidaan teoriassa muuntaa vasemman rekursion välttävään *Greibachin normaalimuotoon*, missä kaikki produktiot ovat muotoa

$$A \to aB_1 \ldots B_k, \qquad k \geq 0,$$

tai $S \to \epsilon$, missä a on päätemerkki, $B_1, \ldots, B_k$ välikkeitä ja S lähtösymboli

4.5.7 *LIITE: FIRST- ja FOLLOW-joukkojen laskeminen

Annettu kielioppi $G = (V, \Sigma, P, S)$.

Ensin lasketaan FIRST-joukot:

1. Asetetaan aluksi kaikille kieliopin päätteille $a \in \Sigma$:

$$\text{FIRST}(a) := \{a\},$$

ja kaikille välikkeille $A \in V - \Sigma$:

$$\begin{aligned} \text{FIRST}(A) \;:=\; & \{a \in \Sigma \mid A \to a\beta \text{ on } G\text{:n produktio}\} \\ & \cup \{\epsilon \mid A \to \epsilon \text{ on } G\text{:n produktio}\}. \end{aligned}$$

2. Käydään sitten kieliopin produktioita läpi jossakin järjestyksessä ja toistetaan, kunnes FIRST-joukot eivät enää kasva: kullekin produktiolle $A \to X_1 \ldots X_k$ asetetaan:

$$\begin{aligned} &\text{FIRST}(A) := \\ &\quad \text{FIRST}(A) \;\cup \\ &\quad \bigcup \{\text{FIRST}(X_i) \mid 1 \leq i \leq k,\ \epsilon \in \text{FIRST}(X_j) \text{ kaikilla } j < k\} \\ &\quad \cup \{\epsilon \mid \epsilon \in \text{FIRST}(X_j) \text{ kaikilla } j = 1, \ldots, k\}. \end{aligned}$$

FOLLOW-joukot määritetään FIRST-joukkojen avulla seuraavasti:

1. Asetetaan aluksi kaikille välikkeille $B \in V - \Sigma^*$:

$$\begin{aligned} &\text{FOLLOW}(B) := \\ &\quad \bigcup \{\text{FIRST}(\beta) - \{\epsilon\} \mid A \to \alpha B \beta \text{ on } G\text{:n produktio}\}, \end{aligned}$$

ja lähtösymbolille S lisäksi:

$$\text{FOLLOW}(S) := \text{FOLLOW}(S) \;\cup\; \{\epsilon\}.$$

2. Sitten toistetaan, kunnes FOLLOW-joukot eivät enää kasva: kullekin produktiolle $A \to \alpha B \beta$, missä $\epsilon \in \text{FIRST}(\beta)$, asetetaan:

$$\text{FOLLOW}(B) := \text{FOLLOW}(B) \cup \text{FOLLOW}(A).$$

4.5.8 LL(1)-kielien vahvemmat sisarukset

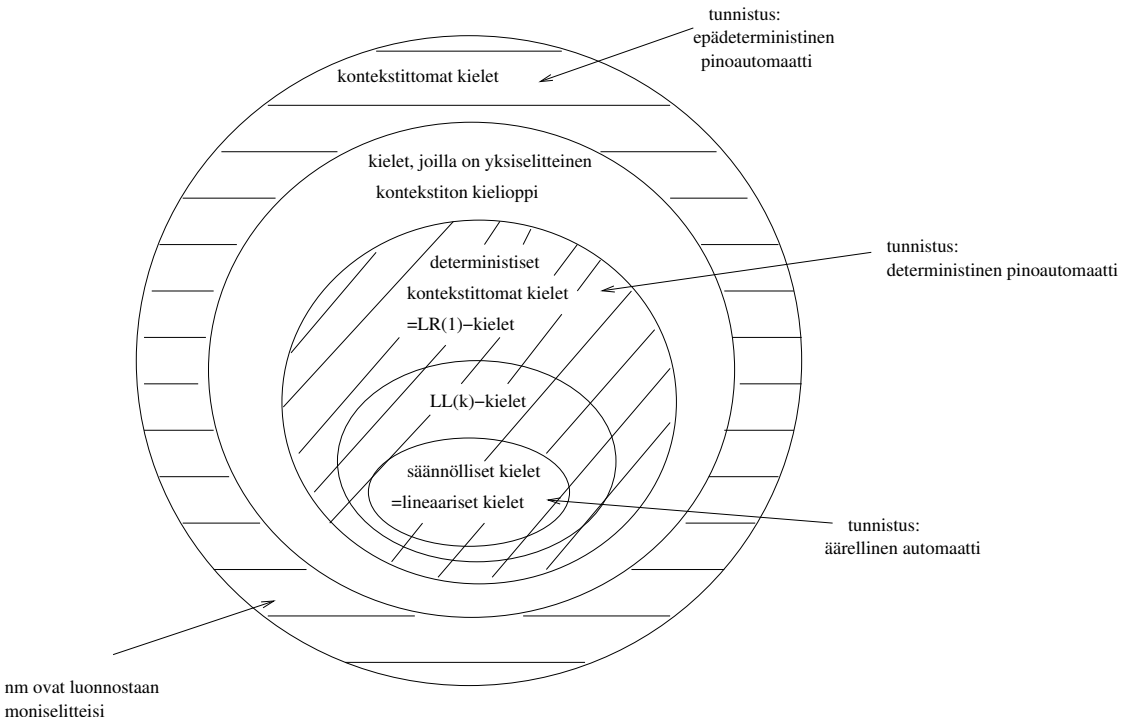
Edellä esitetty rekursiivisen jäsentäjän idea voidaan yleistää tapaukseen, jossa k seuraavaa merkkiä määräävät yksikäsitteisesti, mikä sääntö valitaan kullakin vasemman johdon jäsennysaskeleella. Tällaisia kielioppeja kutsutaan *LL(k)-kieliopeiksi* ("left to right scan, producing left parse with k symbol lookahead").

Vielä laajempi kieliluokka saadaan, kun simuloidaan merkkijon oikeaa johtoa rekursiivisesti. *LR(1)-kieliopissa* ("left to right scan, producing right parse with 1 symbol lookahead") jäsennys etenee oikealta vasemmalle (ts. lavennetaan aina oikeanpuolimmaisin välikesymboli) ja seuraava merkki määrittelee yksikäsitteisesti käytettävän säännön. Vastaavasti *LR(k)-kielissä* seuraavat k merkkiä määrittävät seuraavan johtoaskeleen. Tällaisen alhaalta-ylöspäin (*bottom-up*) [1] etenevän jäsentäjän laatiminen on kuitenkin vaikeampaa kuin LL-kielien edellyttämän ylhäältä alas (*top-down*) [2] etenevän jäsentäjän laatiminen, eikä siihen perehdytä tällä kurssilla. Lisää tietoa löytyy esim. Sudkampin kirjasta.

LR(1)-kielet ovat kuitenkin hyvin tärkeä kieliluokka, sillä ne määrittelevät täsmälleen kaikki *deterministiset* kielet ts. kielet, jotka voidaan tunnistaa deterministisellä pinoautomaatilla!

Huom! On olemassa kieliä, jotka eivät ole deterministisiä, mutta joilla silti on olemassa yksiselitteinen kielioppi. Esimerkiksi pladindromikielelle $L = \{ww^R | w \in \{a, b\}^*\}$ voidaan antaa yksiselitteinen kielioppi, mutta silti sen sanoja ei voida tunnistaa deterministisellä automaatilla. Ongelmana on se, että automaatin täytyy "arvata", milloin on tultu merkkijonon keskikohtaan.

[1] bottom-up-jäsennyksessä jäsennyspuuta muodostetaan alhaalta ylöspäin
[2] top-down-jäsennyksessä jäsennyspuuta ylhäältä alaspäin



Kuva 4.9: Kuva kontekstittomien kielten alaluokkien välisistä suhteista.

4.6 CYK-jäsennysalgoritmi

- Rekursiivisesti etenevä jäsentäminen on selkeä ja tehokas jäsennysmenetelmä LL(1)-kieliopeille: n merkin mittaisen syötemerkkijonon käsittely sujuu ajassa $O(n)$
- Entäpä mielivaltaisen kontekstittoman kieliopin tuottamien merkkijonojen tunnistaminen?
- Ratkaisu 1: muunnetaan kielioppi ensin em. Greibachin normaalimuotoon
  - Tämänmuotoisella kieliopilla millä tahansa n merkin mittaisella lauseella on n:n askeleen johto
  - Merkkijonon x, $|x| = n$, kuuluminen tuotettuun kieleen voidaan testata käymällä läpi kaikki n askelen mittaiset johdot ja katsomalla, tuottaako jokin niistä x:n
  - Tehotonta! (jokaisessa epätriviaalissa kieliopissa on n askelen mittaisia johtoja eksponentiaalinen määrä)

- Ratkaisu 2: *Cocke–Younger–Kasami-* eli *CYK-algoritmi*
  - Toimii ajassa $O(n^3)$, missä n on tutkittavan merkkijonon pituus
  - Kielopin on oltava *Chomskyn normaalimuodossa*

Chomskyn normaalimuoto

Määritelmä: Kontekstiton kielioppi $G = (V, \Sigma, P, S)$ on *Chomskyn normaalimuodossa*, jos

- Välikkeistä enintään S on tyhjentyvä (engl. nullable), eli $S \underset{G}{\Rightarrow}^* \epsilon$
- Muut produktiot ovat muotoa $A \to BC$ tai $A \to a$, missä A, B ja C ovat välikkeitä ja a on päätemerkki
- Lisäksi vaaditaan yksinkertaisuuden vuoksi, että lähtösymboli S ei esiinny minkään produktion oikealla puolella

Esim. seuraava kielioppi on Chomskyn normaalimuodossa:
$S \to AB|\epsilon$
$A \to BA|a$
$B \to b$

Cocke–Younger–Kasami-algoritmi

- Perustuu dynaamiseen ohjelmointiin (välitulosten taulukointiin)
- Ongelma: Olk. $G = (V, \Sigma, P, S)$ Chomskyn normaalimuodossa (ellei ole, muutetaan ensin Chomskyn normaalimuotoon). Onko $x \in L(G)$ eli $S \underset{G}{\Rightarrow}^* x$?

Algoritmi:

1. Jos $x = \epsilon$, niin $x \in L(G) \Leftrightarrow S \to \epsilon$ on G:n produktio
2. Muuten merk. $x = a_1 \dots a_n$ ja tarkastellaan x:n osajonot tuottavien välikkeiden joukkoja

$$N_{i,j} = \{A \in V - \Sigma \mid A \underset{G}{\Rightarrow}^* a_i \dots a_j\}, \qquad 1 \leq i \leq j \leq n$$

3. $x \in L(G) \Leftrightarrow S \in N_{1,n}$

Joukkojen $N_{i,j}$ laskeminen

- muodostetaan kolmiomainen taulukko:

a_1	a_2	a_3	a_4	a_5
$N_{1,1}$				
$N_{1,2}$	$N_{2,2}$			
$N_{1,3}$	$N_{2,3}$	$N_{3,3}$		
$N_{1,4}$	$N_{2,4}$	$N_{3,4}$	$N_{4,4}$	
$N_{1,5}$	$N_{2,5}$	$N_{3,5}$	$N_{4,5}$	$N_{5,5}$

- x-akseli vastaa merkkijonon positiota
- taulukon paikka $N_{i,j} \sim$ joukko välikesymboleita A, joille $A \underset{G}{\Rightarrow}^* a_i a_{i+1} \dots a_j$
- jokainen diagonaali vastaa tietyn pituisia x:n osajonoja
  - 1. diagonaali $\sim$ välikkeet, jotka tuottavat yhden merkin osajonot
  - 2. diagonaali $\sim$ välikkeet, jotka tuottavat kahden merkin osajonot jne.

(i) Lasketaan ensin for $i = 1$ to n:

$$N_{i,i} := \{A \in V - \Sigma \mid A \to a_i \text{ on } G\text{:n produktio}\};$$

(ii) Lasketaan sitten induktiivisesti

for $k = 1$ to $n - 1$
  for $i = 1$ to $n - k$:

$$N_{i,i+k} := \{A \in V - \Sigma \mid \text{jollakin } j,\ i \le j < i + k, \text{ on välikkeet } B \in N_{ij} \text{ ja } C \in N_{j+1,i+k} \text{ s.e. } A \to BC \text{ on } G\text{:n produktio}\}$$

- Huom! Tunnetaan jo kaikki lyhyemmät osajonot ja kaikki x:n kelvolliset prefiksit ja suffiksit
- Johdon $A \underset{G}{\Rightarrow}^* a_i a_{i+1} \dots a_j$ täytyy alkaa askeleella $A \Rightarrow BC$, missä B:stä voidaan johtaa $a_i \dots a_j$:n prefiksi, olk. $B \underset{G}{\Rightarrow}^* a_i a_{i+1} \dots a_l$, ja C:stä voidaan johtaa loput:
  $C \underset{G}{\Rightarrow}^* a_{l+1} a_{l+2} \dots a_j$
- Esim. $N_{3,7}$:ta varten tarkistellaan kaikkia pareja
  $(N_{3,3}, N_{4,7}), (N_{3,4}, N_{5,7}), (N_{3,5}, N_{6,7}), (N_{3,6}, N_{7,7})$

Esim. Sovelletaan CYK:iä Chomskyn normaalimuotoiseen kielioppiin

$$\begin{array}{lcll} S & \to & AB & | \quad BC \\ A & \to & BA & | \quad a \\ B & \to & CC & | \quad b \\ C & \to & AB & | \quad a \end{array}$$

syötemerkkijonolla $x = baaba$:

	$i \to$			
$1 : b$	$2 : a$	$3 : a$	$4 : b$	$5 : a$
B				
S, A	A, C			
$\emptyset$	B	A, C		
$\emptyset$	B	S, C	B	
S, A, C	S, A, C	B	S, A	A, C

Lähtösymboli S kuuluu joukkoon $N_{1,5}$, joten x kuuluu kieliopin tuottamaan kieleen