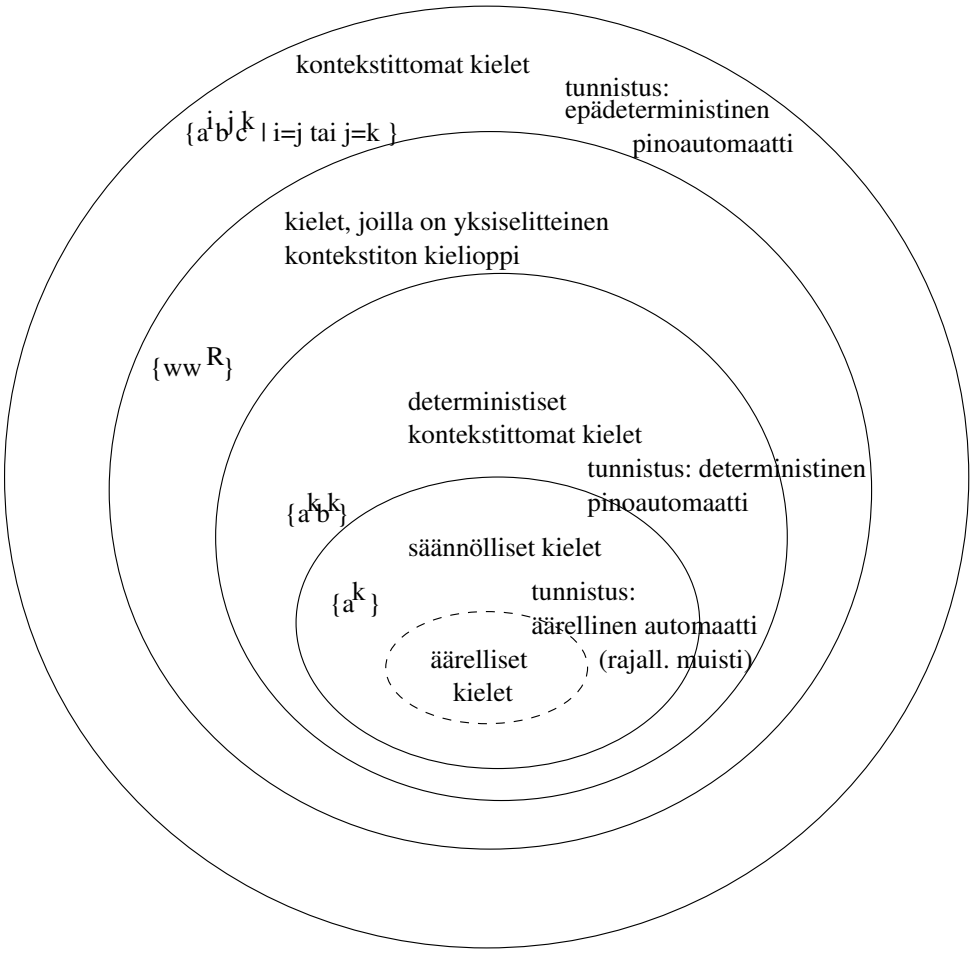


4.4 Kielioppien jäsennysongelma

Ratkaistava tehtävä:

"Annettu kontekstiton kielioppi G ja merkkijono x. Onko $x \in L(G)$?"

- Esim. Kuuluuko virke "jänis joka pelkäsi arkaa peikkoa metsästi suurta sutta" kieleen L_{rel}?
- Ratkaisumenetelmä: *jäsennysalgoritmi*
- Useita vaihtoehtoisia menetelmiä, erityisesti kun G on jotain rajoitettua (käytännössä esiintyvää) muotoa.

- Tarkastellaan ensin jäsentämiseen liittyviä peruskäsitteitä

4.4.1 Johdot

- Olk. merkkijono $\gamma \in V^*$ kieliopin $G = (V, \Sigma, P, S)$ lausejohdos
- γ:n *johdoksi* G:ssä kutsutaan lähtösymbolista S merkkijonoon γ johtavaa suorien johtojen jonoa

$$S = \gamma_0 \Rightarrow \gamma_1 \Rightarrow \cdots \Rightarrow \gamma_n = \gamma$$

- Johdon *pituus* on siihen kuuluvien suorien johtojen määrä (edellä n)
- Esim. lauseen $a + a$ johtoja kieliopissa G_{expr}:

$$\begin{array}{lllllllll}
\text{(i)} & E & \Rightarrow & E+T & \Rightarrow & T+T & \Rightarrow & F+T \\
 & & \Rightarrow & a+T & \Rightarrow & a+F & \Rightarrow & a+a \\
\text{(ii)} & E & \Rightarrow & E+T & \Rightarrow & E+F & \Rightarrow & T+F \\
 & & \Rightarrow & F+F & \Rightarrow & F+a & \Rightarrow & a+a \\
\text{(iii)} & E & \Rightarrow & E+T & \Rightarrow & E+F & \Rightarrow & E+a \\
 & & \Rightarrow & T+a & \Rightarrow & F+a & \Rightarrow & a+a
\end{array}$$

- Johto $\gamma \Rightarrow^* \gamma'$ on
  1. *vasen johto*, merk.

     $$\gamma \underset{\text{lm}}{\Rightarrow}{}^* \gamma',$$

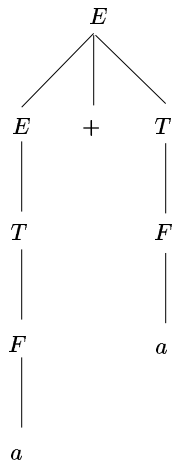
     jos kussakin johtoaskelessa on produktiota sovellettu merkkijonon vasemmanpuoleisimpaan välikkeeseen (edellä johto (i))
  2. *oikea johto*, merk.

     $$\gamma \underset{\text{rm}}{\Rightarrow}{}^* \gamma',$$

     jos — ” — oikeanpuoleiseen välikkeeseen (edellä (iii))
- Suoria vasempia ja oikeita johtoaskelia merkitään $\gamma \underset{\text{lm}}{\Rightarrow} \gamma'$ ja $\gamma \underset{\text{rm}}{\Rightarrow} \gamma'$

4.4.2 Jäsennyspuut

- Suurin osa vaihtoehtoisten johtotapojen eroista muodostuu vain välikkeiden laventamisesta eri järjestyksessä (esim. em. johdot (i) – (iii) ovat kaikki "pohjimmiltaan" samanlaisia)
- Lausejohdoksen *jäsennyspuu* (syntaksipuu, johtopuu) (engl. parse tree, syntax tree, derivation tree) = esitystapa, jossa nämä epäoleelliset erot on abstrahoitu pois
- Jäsennyspuu kertoo ainoastaan, *miten* välikkeet on lavennettu, ei *missä järjestyksessä* lavennukset on tehty
- Esim. kaikkia kolmea em. johtoa vastaa sama jäsennyspuu:



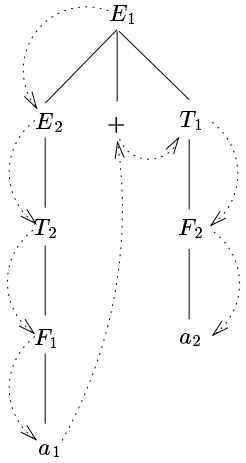
Kuva 4.6: Lauseen $a + a$ jäsennyspuu kieliopissa G_{expr}.

Määritelmä: Olkoon $G = (V, \Sigma, P, S)$ kontekstiton kielioppi. Kieliopin G mukainen *jäsennyspuu* on järjestetty puu, jolla on seuraavat ominaisuudet:

(i) puun solmut on nimetty joukon $V \cup \{\epsilon\}$ alkioilla siten, että sisäsolmujen nimet ovat välikkeitä (so. joukosta $N = V - \Sigma$) ja juurisolmun nimenä on lähtösymboli S;

(ii) jos A on puun jonkin sisäsolmun nimi, ja $X_1, \ldots, X_k$ ovat sen jälkeläisten nimet järjestyksessä, niin
$A \to X_1 \ldots X_k$ on G:n produktio

- Jäsennyspuun τ *tuotos* = merkkijono, joka saadaan liittämällä yhteen sen lehtisolmujen nimet esijärjestyksessä ("vasemmalta oikealle")



$$E_1E_2T_2F_1a_1 + T_1F_2a_2$$

$$E \underset{\text{lm}}{\Rightarrow} E + T \underset{\text{lm}}{\Rightarrow} T + T \underset{\text{lm}}{\Rightarrow} F + T \underset{\text{lm}}{\Rightarrow} a + T \underset{\text{lm}}{\Rightarrow} a + F \underset{\text{lm}}{\Rightarrow} a + a$$

Kuva 4.7: Vasemman johdon muodostaminen jäsennyspuusta.

4.4.3 Jäsennyspuun muodostaminen

- Johtoa

$$S = \gamma_0 \Rightarrow \gamma_1 \Rightarrow \cdots \Rightarrow \gamma_n = \gamma$$

vastaava jäsennyspuu muodostetaan seuraavasti:

(i) puun juuren nimeksi tulee S; jos $n = 0$, niin puussa ei ole muita solmuja; muuten

(ii) jos ensimmäisessä johtoaskelessa on sovellettu produktiota $S \to X_1X_2 \ldots X_k$, niin juurelle tulee k jälkeläissolmua, joiden nimet vasemmalta oikealle ovat $X_1, X_2, \ldots, X_k$;

(iii) jos seuraavassa askelessa on sovellettu produktiota $X_i \to Y_1Y_2 \ldots Y_l$, niin juuren i:nnelle jälkeläissolmulle tulee l jälkeläistä, joiden nimet vasemmalta oikealle ovat $Y_1, Y_2, \ldots, Y_l$; ja niin edelleen

- Ts. jos τ on jotakin johtoa $S \Rightarrow^* \gamma$ vastaava jäsennyspuu, niin τ:n tuotos on γ
- Johdon muodostaminen: Olk. τ kieliopin G mukainen jäsennyspuu, jonka tuotos on päätemerkkijono x.
  - τ:sta saadaan vasen johto x:lle käymällä puun solmut läpi esijärjestyksessä ("ylhäältä alas, vasemmalta oikealle") ja laventamalla vastaan tulevat välikkeet järjestyksessä puun osoittamalla tavalla

  - Oikea johto saadaan käymällä puu läpi käänteisessä esijärjestyksessä ("ylhäältä alas, oikealta vasemmalle")

- Jos muodostetaan annetusta vasemmasta (oikeasta) johdosta $S \underset{\text{lm}}{\Rightarrow}^* x$ ($S \underset{\text{rm}}{\Rightarrow}^* x$) ensin jäsennyspuu em. tavalla, ja sitten jäsennyspuusta vasen (oikea) johto, saadaan takaisin alkuperäinen johto

Lause: Olk. $G = (V, \Sigma, P, S)$ kontekstiton kielioppi. Tällöin:

(i) jokaisella G:n lausejohdoksella γ on G:n mukainen jäsennyspuu τ, jonka tuotos on γ;

(ii) jokaista G:n mukaista jäsennyspuuta τ, jonka tuotos on päätemerkkijono x, vastaavat yksikäsitteiset vasen ja oikea johto $S \underset{\text{lm}}{\Rightarrow}^* x$ ja $S \underset{\text{rm}}{\Rightarrow}^* x$

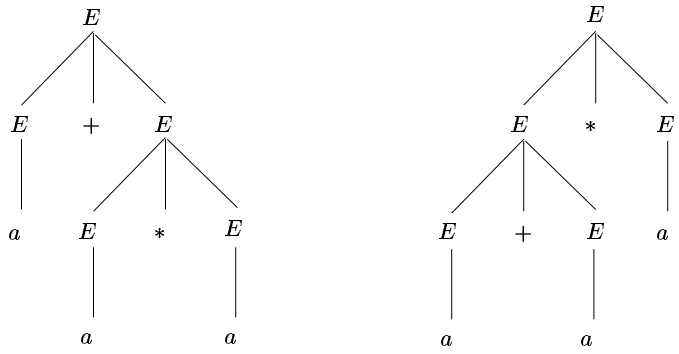
Seuraus: Jokaisella G:n lauseella on vasen ja oikea johto

- Ts. kontekstittoman kieliopin tuottamien lauseiden jäsennyspuut, vasemmat ja oikeat johdot vastaavat yksikäsitteisesti toisiaan

- Jäsennysongelman ratkaisuksi katsotaan usein päätösongelman "Onko $x \in L(G)$?" ratkaisemisen lisäksi jonkin näistä jäsennysesityksistä tuottaminen x:lle

4.4.4 Kieliopin moniselitteisyys

- Lauseella voi olla kieliopissa useita jäsennyksiä (esim. lause $a+a*a$ kieliopissa G'_{expr})



Kuva 4.8: Lauseen $a + a * a$ kaksi erilaista jäsennystä.

- 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

- 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':

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

- G' 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
- Kielioppia, jolla on tämä ominaisuus, sanotaan *LL(1)-kieliopiksi* ("left to right scan, producing left parse with 1 symbol lookahead" – vast LL(k)- ja LR(k)-kieliopit)

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:

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

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:

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

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:

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

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 → ω1 | ... | ωn*/
begin
    if (next in [a11, ..., a1m1])          /*FIRST({ω1}FOLLOW(A)) = {a11, ..., a1m1}*/
    begin /*produktio A → ω1*/
        parse(ω1);
    end
    else
```

```
    ⋮
    if (next in [a_{n1}, ..., a_{nm_n}])        /*FIRST({ω_n}FOLLOW(A)) = {a_{n1}, ..., a_{nm_n}}*/
    begin /*produktio A → ω_n*/
        parse(ω_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)

4.5.5 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

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

  ei voi olla LL(1)-muotoinen

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

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

  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

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

  produktioilla

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

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

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

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

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

4.5.6 *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:

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

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:

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