

Süntaksanalüüs

Ennustav parsimine

Rekursiivse laskumise meetod

Tabeljuhitav LL(1) parser

Vasakfaktoreerimine

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:

$$\begin{aligned} \text{Nullable}(\varepsilon) &= \text{true} \\ \text{Nullable}(c) &= \text{false} \\ \text{Nullable}(\alpha\beta) &= \text{Nullable}(\alpha) \wedge \text{Nullable}(\beta) \\ \text{Nullable}(A) &= \text{Nullable}(\alpha_1) \vee \dots \vee \text{Nullable}(\alpha_n) \end{aligned}$$

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 parempoolne;
- $\text{First}(\beta) \subseteq \text{Follow}(A)$;
- $\text{Follow}(B) \subseteq \text{Follow}(A)$, kui $\text{Nullable}(\beta)$.

Ennustav parsimine

Näide:

$$S \rightarrow A B C$$

$$A \rightarrow a A \mid \varepsilon$$

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

$$C \rightarrow c \mid d$$

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

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

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

$$\textit{First}(S) = \textit{First}(A B C)$$

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

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

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

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

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

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

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

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

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

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

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

Ennustav parsimine

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

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

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

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

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 $A \rightarrow a B \mid b C A \mid D E$
ja $la(A \rightarrow D E) = \{c\}$

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

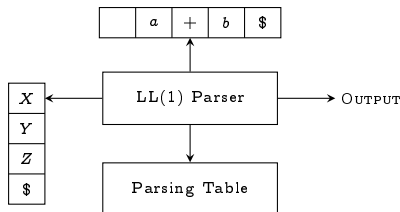
Ennustav parsimine

Näide: olgu A produktsioonireeglid $A \rightarrow a B \mid b C A \mid D E$
ja $la(A \rightarrow D E) = \{c\}$

```
parseA() {  
  if token = a then {  
    token := nextWord(); t1 := parseB();  
    return(mkTreeRule1(t1));  
  } else if token = b then {  
    token := nextword(); t1 := parseC(); t2 := parseA();  
    return(mkTreeRule2(t1, t2));  
  } else if token = c then {  
    t1 := parseD(); t2 := parseE();  
    return(mkTreeRule3(t1, t2));  
  } else {  
    error();  
  }  
}
```

Ennustav parsimine

- Automaatselt genereeritavate LL(1) parserite korral kasutatakse tavaliselt tabeljuhitavat LL(1) parsimist:
 - moodustatakse maatriks M , kus read ja veerud on vastavalt indekseeritud mitteterminaalide ja terminalidega;
 - tabeli pesades on produktsioonireglid, mida antud mitteterminaali 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();  
}
```

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 prefixsit.