

Operator Precedence Languages

Subtitle

Agenda

1. Introduction
2. Preliminaries
3. Operator Precedence Languages
4. (Closure-)Properties

Introduction

1.

Definition kontextfreie Grammatik

- ▶ Eine kontextfreie Grammatik ist ein 4-Tupel $G = (N, \Sigma, P, S)$

Definition kontextfreie Grammatik

- ▶ Eine kontextfreie Grammatik ist ein 4-Tupel $G = (N, \Sigma, P, S)$
 - ▶ N ist die Menge der Nichtterminale

Definition kontextfreie Grammatik

- ▶ Eine kontextfreie Grammatik ist ein 4-Tupel $G = (N, \Sigma, P, S)$
 - ▶ N ist die Menge der Nichtterminale
 - ▶ Σ ist die Menge der Terminalsymbole

Definition kontextfreie Grammatik

- ▶ Eine kontextfreie Grammatik ist ein 4-Tupel $G = (N, \Sigma, P, S)$
 - ▶ N ist die Menge der Nichtterminale
 - ▶ Σ ist die Menge der Terminalsymbole
 - ▶ P sind die Produktionsregeln

Definition kontextfreie Grammatik

- ▶ Eine kontextfreie Grammatik ist ein 4-Tupel $G = (N, \Sigma, P, S)$
 - ▶ N ist die Menge der Nichtterminale
 - ▶ Σ ist die Menge der Terminalsymbole
 - ▶ P sind die Produktionsregeln
 - ▶ $S \in N$ ist das Startsymbol

Namenskonventionen 1

1. $a, b, \dots \in \Sigma$ sind einzelne Terminalsymbole
2. $v, w, \dots \in \Sigma^*$ sind beliebige Terminalstrings
3. $A, B, \dots \in N$ sind einzelne Nichtterminale
4. $\alpha, \beta, \dots \in (\Sigma \cup N)^*$ sind beliebige Reststrings
5. $A \rightarrow \epsilon$ ist die *leere Regel*
6. Eine *umbenennende* Regel hat nur ein Nichtterminal als rechte Seite ($A \rightarrow B$)

Namenskonventionen 2

1. Eine Regel ist in *Operatorform*, wenn die rechte Seite keine benachbarten Nichtterminale hat
2. Jede kontextfreie Grammatik kann in eine äquivalente *Operatorgrammatik (OG)* umgewandelt werden

Operator Precedence Grammar

- ▶ Wird häufig auch *Floyd Grammar* genannt nach ihrem Erfinder

Operator Precedence Grammar

- ▶ Wird häufig auch *Floyd Grammar* genannt nach ihrem Erfinder
- ▶ Definition: *Linke und Rechte Terminalmenge*

$$\mathcal{L}_G(A) = \{a \in \Sigma \mid A \xRightarrow{*} Ba\alpha\}$$

$$\mathcal{R}_G(A) = \{a \in \Sigma \mid A \xRightarrow{*} \alpha aB\}$$

Operator Precedence Grammar

- ▶ Wird häufig auch *Floyd Grammar* genannt nach ihrem Erfinder
- ▶ Definition: *Linke und Rechte Terminalmenge*

$$\mathcal{L}_G(A) = \{a \in \Sigma \mid A \xRightarrow{*} Ba\alpha\}$$

$$\mathcal{R}_G(A) = \{a \in \Sigma \mid A \xRightarrow{*} \alpha aB\}$$

- ▶ Es werden drei binäre Operator Precedence Relationen definiert:

Operator Precedence Grammar

- ▶ Wird häufig auch *Floyd Grammar* genannt nach ihrem Erfinder
- ▶ Definition: *Linke und Rechte Terminalmenge*

$$\mathcal{L}_G(A) = \{a \in \Sigma \mid A \xRightarrow{*} Ba\alpha\}$$

$$\mathcal{R}_G(A) = \{a \in \Sigma \mid A \xRightarrow{*} \alpha aB\}$$
- ▶ Es werden drei binäre Operator Precedence Relationen definiert:
- ▶ equal in precedence: $a \doteq b \Leftrightarrow \exists A \rightarrow \alpha aBb\beta, B \in N \cup \{\epsilon\}$
takes precedence: $a \succ b \Leftrightarrow \exists A \rightarrow \alpha Db\beta, D \in N \text{ and } a \in \mathcal{R}_G(D)$
yields precedence: $a \prec b \Leftrightarrow \exists A \rightarrow \alpha aD\beta, D \in N \text{ and } b \in \mathcal{L}_G(D)$

Operator Precedence Languages

- ▶ Ein nichtdeterministischer Operator Precedence Automat ist ein 6-Tupel $A = (\Sigma, M, Q, I, F, \delta)$

Operator Precedence Languages

- ▶ Ein nichtdeterministischer Operator Precedence Automat ist ein 6-Tupel $A = (\Sigma, M, Q, I, F, \delta)$
 1. (Σ, M) ist ein OP alphabet

Operator Precedence Languages

- ▶ Ein nichtdeterministischer Operator Precedence Automat ist ein 6-Tupel $A = (\Sigma, M, Q, I, F, \delta)$
 1. (Σ, M) ist ein OP alphabet
 2. Q ist die Menge der Zustände

Operator Precedence Languages

- ▶ Ein nichtdeterministischer Operator Precedence Automat ist ein 6-Tupel $A = (\Sigma, M, Q, I, F, \delta)$
 1. (Σ, M) ist ein OP alphabet
 2. Q ist die Menge der Zustände
 3. $I \subseteq Q$ ist die Menge der Startzustände

Operator Precedence Languages

- ▶ Ein nichtdeterministischer Operator Precedence Automat ist ein 6-Tupel $A = (\Sigma, M, Q, I, F, \delta)$
 1. (Σ, M) ist ein OP alphabet
 2. Q ist die Menge der Zustände
 3. $I \subseteq Q$ ist die Menge der Startzustände
 4. $F \subseteq Q$ ist die Menge der finalen Zustände

Operator Precedence Languages

- ▶ Ein nichtdeterministischer Operator Precedence Automat ist ein 6-Tupel $A = (\Sigma, M, Q, I, F, \delta)$
 1. (Σ, M) ist ein OP alphabet
 2. Q ist die Menge der Zustände
 3. $I \subseteq Q$ ist die Menge der Startzustände
 4. $F \subseteq Q$ ist die Menge der finalen Zustände
 5. δ ist die Übergangsfunktion, die aus drei Teilen besteht:
 $\delta_{\text{shift}} : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ $\delta_{\text{push}} : Q \times \Sigma \rightarrow \mathcal{P}(Q)$ $\delta_{\text{pop}} : Q \times Q \rightarrow \mathcal{P}(Q)$

(Closure-)Properties

- ▶ OPLs sind eine große Subklasse der kontextfreien Sprachen, die Abgeschlossenheitseigenschaften von regulären Sprachen genießt

(Closure-)Properties

- ▶ OPLs sind eine große Subklasse der kontextfreien Sprachen, die Abgeschlossenheitseigenschaften von regulären Sprachen genießt
- ▶ Abgeschlossen unter Vereinigung, Schnitt, Komplement, Konkatenation und Kleene-*

(Closure-)Properties

- ▶ OPLs sind eine große Subklasse der kontextfreien Sprachen, die Abgeschlossenheitseigenschaften von regulären Sprachen genießt
- ▶ Abgeschlossen unter Vereinigung, Schnitt, Komplement, Konkatenation und Kleene-*
- ▶ Das Leereproblem ist in PTIME lösbar, da OPLs Subklasse von kfG

(Closure-)Properties

- ▶ OPLs sind eine große Subklasse der kontextfreien Sprachen, die Abgeschlossenheitseigenschaften von regulären Sprachen genießt
- ▶ Abgeschlossen unter Vereinigung, Schnitt, Komplement, Konkatenation und Kleene-*
- ▶ Das Leereproblem ist in PTIME lösbar, da OPLs Subklasse von kfG
- ▶ Visibly Pushdown Sprachen sind in der Klasse der OPLs enthalten

Quellen