WWU
MÜNSTER

Operator Präzedenz Sprachen

und ihre Automaten

wissen.leben
WWU Münster

Jonas Kremer

10. Dezember 2018

Agenda

1. Introduction
2. Definitionen und Namenskonventionen
3. Operator Precedence Languages
4. Automaten Beispiel
5. (Closure-)Properties

Introduction

1. Operator Precedence Languages (OPL) sind eine Subklasse der kontextfreien Sprachen

Introduction

1. Operator Precedence Languages (OPL) sind eine Subklasse der kontextfreien Sprachen
2. Sie wurden 1963 von Robert W. Floyd vorgestellt

Introduction

1. Operator Precedence Languages (OPL) sind eine Subklasse der kontextfreien Sprachen
2. Sie wurden 1963 von Robert W. Floyd vorgestellt
3. Größte bekannte Klasse von kontextfreien Sprache, die Abschlusseigenschaften von regulären Sprachen hat

Introduction

1. Operator Precedence Languages (OPL) sind eine Subklasse der kontextfreien Sprachen
2. Sie wurden 1963 von Robert W. Floyd vorgestellt
3. Größte bekannte Klasse von kontextfreien Sprache, die Abschlusseigenschaften von regulären Sprachen hat
4. Als Beispiel dient eine einfache Sprache für arithmetische Ausdrücke
   zB. $n + n \times (n + n)$

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 $G = (N, \Sigma, P, S)$
  1. N ist die Menge der Nichtterminale

Definition kontextfreie Grammatik

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

Definition kontextfreie Grammatik

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

Definition kontextfreie Grammatik

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

Definition kontextfreie Grammatik

- Eine kontextfreie Grammatik ist ein 4-Tupel $G = (N, \Sigma, P, S)$
  1. N ist die Menge der Nichtterminale
  2. Σ ist die Menge der Terminalsymbole
  3. P sind die Produktionsregeln
  4. S $\in$ N ist das Startsymbol
- Produktionen haben die Form $A \rightarrow \beta$, wobei $\beta \in (\Sigma \cup N) \cup \epsilon$

Definition kontextfreie Grammatik

- Eine kontextfreie Grammatik ist ein 4-Tupel $G = (N, \Sigma, P, S)$
  1. N ist die Menge der Nichtterminale
  2. Σ ist die Menge der Terminalsymbole
  3. P sind die Produktionsregeln
  4. S $\in$ N ist das Startsymbol
- Produktionen haben die Form $A \rightarrow \beta$, wobei $\beta \in (\Sigma \cup N) \cup \epsilon$
- Ableitungen werden mit $\Rightarrow$ bzw. $\overset{*}{\Rightarrow}$ beschrieben

Namenskonventionen 1

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

Namenskonventionen 2

- Eine Grammatik ist *reduziert*, wenn jede Regel benutzt werden kann um ein Wort aus Σ^* zu erzeugen

Namenskonventionen 2

- Eine Grammatik ist *reduziert*, wenn jede Regel benutzt werden kann um ein Wort aus Σ^* zu erzeugen
- Eine Grammatik ist *invertierbar*, wenn es keine identischen rechten Seiten von Produktionsregeln gibt

Namenskonventionen 2

- Eine Grammatik ist *reduziert*, wenn jede Regel benutzt werden kann um ein Wort aus Σ^* zu erzeugen
- Eine Grammatik ist *invertierbar*, wenn es keine identischen rechten Seiten von Produktionsregeln gibt
- Eine Regel ist in *Operatorform*, wenn die rechte Seite keine benachbarten Nichtterminale hat

Namenskonventionen 2

- Eine Grammatik ist *reduziert*, wenn jede Regel benutzt werden kann um ein Wort aus Σ^* zu erzeugen
- Eine Grammatik ist *invertierbar*, wenn es keine identischen rechten Seiten von Produktionsregeln gibt
- Eine Regel ist in *Operatorform*, wenn die rechte Seite keine benachbarten Nichtterminale hat
- Jede kontextfreie Grammatik kann in eine äquivalente *Operatorgrammatik (OG)* umgewandelt werden

Operator Precedence Grammar

- Definition: *Linke und Rechte Terminalmenge*
  $\mathcal{L}_G(A) = \{a \in \Sigma | A \overset{*}{\Rightarrow} Ba\alpha\}$
  $\mathcal{R}_G(A) = \{a \in \Sigma | A \overset{*}{\Rightarrow} \alpha aB\}$

Operator Precedence Grammar

- Definition: *Linke und Rechte Terminalmenge*
  $\mathcal{L}_G(A) = \{a \in \Sigma | A \overset{*}{\Rightarrow} Ba\alpha\}$
  $\mathcal{R}_G(A) = \{a \in \Sigma | A \overset{*}{\Rightarrow} \alpha aB\}$
- Es werden drei binäre Operator Precedence Relationen definiert:

Operator Precedence Grammar

- Definition: *Linke und Rechte Terminalmenge*
  $\mathcal{L}_G(A) = \{a \in \Sigma | A \overset{*}{\Rightarrow} Ba\alpha\}$
  $\mathcal{R}_G(A) = \{a \in \Sigma | A \overset{*}{\Rightarrow} \alpha aB\}$
- Es werden drei binäre Operator Precedence Relationen definiert:
- equal in precedence: $a \doteq b \Leftrightarrow \exists A \to \alpha aBb\beta, B \in N \cup \{\epsilon\}$
  takes precedence: $a \gtrdot b \Leftrightarrow \exists A \to \alpha Db\beta, D \in N$ and $a \in \mathcal{R}_G(D)$
  yields precedence: $a \lessdot b \Leftrightarrow \exists A \to \alpha aD\beta, D \in N$ and $b \in \mathcal{L}_G(D)$

Operator Precedence Grammar

- Eine Operator Precedence Matrix (OPM) M ist eine $|\Sigma| \times |\Sigma|$ Matrix, die für jedes Paar (a,b) die Precedence Relation speichert

Operator Precedence Grammar

- Eine Operator Precedence Matrix (OPM) M ist eine $|\Sigma| \times |\Sigma|$ Matrix, die für jedes Paar (a,b) die Precedence Relation speichert
- Eine Operator Grammatik ist eine Operator Precedence Grammatik, wenn M=OPM(G) *konfliktfrei* ist

Operator Precedence Grammar

- Eine Operator Precedence Matrix (OPM) M ist eine $|\Sigma| \times |\Sigma|$ Matrix, die für jedes Paar (a,b) die Precedence Relation speichert
- Eine Operator Grammatik ist eine Operator Precedence Grammatik, wenn M=OPM(G) *konfliktfrei* ist
- Eine OPG ist in *Fischer Normalform*, wenn sie invertierbar ist und keine leeren (ausser Startsymbol) oder umbenennenden Regeln hat

Operator Precedence Grammar

- Eine Operator Precedence Matrix (OPM) M ist eine $|\Sigma| \times |\Sigma|$ Matrix, die für jedes Paar (a,b) die Precedence Relation speichert
- Eine Operator Grammatik ist eine Operator Precedence Grammatik, wenn M=OPM(G) *konfliktfrei* ist
- Eine OPG ist in *Fischer Normalform*, wenn sie invertierbar ist und keine leeren (ausser Startsymbol) oder umbenennenden Regeln hat
- Zusätzliches Symbol $\# \notin \Sigma$ um das Ende eines Strings zu markieren
  Alle anderen Symbole übernehmen Precedence über $\#$

Operator Precedence Grammar

- Eine Operator Precedence Matrix (OPM) M ist eine $|\Sigma| \times |\Sigma|$ Matrix, die für jedes Paar (a,b) die Precedence Relation speichert
- Eine Operator Grammatik ist eine Operator Precedence Grammatik, wenn M=OPM(G) *konfliktfrei* ist
- Eine OPG ist in *Fischer Normalform*, wenn sie invertierbar ist und keine leeren (ausser Startsymbol) oder umbenennenden Regeln hat
- Zusätzliches Symbol $\# \notin \Sigma$ um das Ende eines Strings zu markieren
  Alle anderen Symbole übernehmen Precedence über $\#$
- Ein Operator Precedence Alphabet ist ein Paar (Σ, M) mit der konfliktfreien OPM $M = |\Sigma \cup \{\#\}|^2$

Operator Precedence Automata

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

Operator Precedence Automata

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

Operator Precedence Automata

- Ein nichtdeterministischer Operator Precedence Automat (OPA) 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 Automata

- Ein nichtdeterministischer Operator Precedence Automat (OPA) 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 Automata

- Ein nichtdeterministischer Operator Precedence Automat (OPA) 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 Automata

- Ein nichtdeterministischer Operator Precedence Automat (OPA) 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:
     $\delta_{\text{shift}} : Q \times \Sigma \to \mathcal{P}(Q)$ $\delta_{\text{push}} : Q \times \Sigma \to \mathcal{P}(Q)$ $\delta_{\text{pop}} : Q \times Q \to \mathcal{P}(Q)$

Operator Precedence Automata

- Ein nichtdeterministischer Operator Precedence Automat (OPA) 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:
     $\delta_{\text{shift}} : Q \times \Sigma \to \mathcal{P}(Q)$ $\delta_{\text{push}} : Q \times \Sigma \to \mathcal{P}(Q)$ $\delta_{\text{pop}} : Q \times Q \to \mathcal{P}(Q)$
- Weiterhin wird das Stackalphabet definiert als $\Gamma = (\Sigma \times Q)$ mit $[a, q] \in \Gamma$ und dem Symbol für den leeren Stack $\perp$

Operator Precedence Automata

- Ein nichtdeterministischer Operator Precedence Automat (OPA) 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:
     $\delta_{\text{shift}} : Q \times \Sigma \to \mathcal{P}(Q)$ $\delta_{\text{push}} : Q \times \Sigma \to \mathcal{P}(Q)$ $\delta_{\text{pop}} : Q \times Q \to \mathcal{P}(Q)$
- Weiterhin wird das Stackalphabet definiert als $\Gamma = (\Sigma \times Q)$ mit $[a, q] \in \Gamma$ und dem Symbol für den leeren Stack $\perp$
- Der Stack Π ist ein String $\perp \Gamma^*$
  Bsp: $\perp [+, q_1] [n, q_0]$

Operator Precedence Automata

- 1 Eine *Konfiguration* eines OPA ist ein Tripel $C = (\Pi, q, w)$ mit dem Stack Π, dem aktuellen Zustand q und der Eingabe w
- Eine Berechnung des Automaten ist eine endliche Folge von *Transitionen (Moves)* $C_1 \vdash C_2$
- Die Sprache die ein OPA A akzeptiert wird definiert als: L(A)= $\{x | (\bot, q_I, x\#) \vdash^* (\bot, q_F, \#), q_I \in I, q_F \in F\}$

OPA Transitionen

- *push move:* (Normaler Pfeil)
  if $sym(\Pi) \lessdot a$ then $(\Pi, p, ax) \vdash (\Pi\,[a, p]\,, q, x)$ mit $q \in \delta_{\text{push}}(p, a)$
- *shift move:* (Gestrichelter Pfeil)
  if $a \doteq b$ then $(\Pi\,[a, p]\,, q, bx) \vdash (\Pi\,[b, p]\,, r, x)$ mit $r \in \delta_{\text{shift}}(q, b)$
- *pop move:* (Doppelter Pfeil)
  if $a \gtrdot b$ then $(\Pi\,[a, p]\,, q, bx) \vdash (\Pi, r, bx)$ mit $r \in \delta_{\text{pop}}(q, p)$

Stack	Zustand	Eingabe
$\perp$	q_0	$n + n \times (n + n)\#$
$\perp\,[n, q_0]$	q_1	$+n \times (n + n)\#$
$\perp$	q_1	$+n \times (n + n)\#$
$\perp\,[+, q_1]$	q_0	$n \times (n + n)\#$
$\perp\,[+, q_1]\,[n, q_0]$	q_1	$\times(n + n)\#$
$\perp\,[+, q_1]$	q_1	$\times(n + n)\#$
$\perp\,[+, q_1]\,[\times, q_0]$	q_1	$(n + n)\#$
$\perp\,[+, q_1]\,[\times, q_1]\,[(, q_0]$	q_2	$n + n)\#$
$\perp\,[+, q_1]\,[\times, q_1]\,[(, q_0]\,[n, q_2]$	q_3	$+n)\#$
$\perp\,[+, q_1]\,[\times, q_1]\,[(, q_0]$	q_3	$+n)\#$
$\perp\,[+, q_1]\,[\times, q_1]\,[(, q_0]\,[+, q_3]$	q_2	$n)\#$
$\perp\,[+, q_1]\,[\times, q_1]\,[(, q_0]\,[+, q_3]\,[n, q_2]$	q_3	$)\#$
$\perp\,[+, q_1]\,[\times, q_1]\,[(, q_0]\,[+, q_3]$	q_3	$)\#$
$\perp\,[+, q_1]\,[\times, q_1]\,[(, q_0]$	q_3	$)\#$
$\perp\,[+, q_1]\,[\times, q_1]\,[), q_0]$	q_3	$\#$
$\perp\,[+, q_1]\,[\times, q_1]$	q_3	$\#$
$\perp\,[+, q_1]$	q_3	$\#$
$\perp$	q_3	$\#$

(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