Operator Precedence Sprachen und ihre Automaten

Jonas Kremer

7. Februar 2019

Inhaltsverzeichnis

1 Motivation

2 Vorbereitung und Definitionen

In diesem Abschnitt geht es um die Definitionen und Namenskonventionen, die für Operatorpräzedenzsprachen benötigt werden. Zunächst kommt eine kurze Definition von kontextfreien Grammatiken, gefolgt von Definitionen und Einschränkungen für Operatorpräzedenzgrammatiken (OPG). Im Anschluss wird die Klasse der Operatorpräzedenzautomaten (OPA) eingeführt, sowohl deterministisch als auch nichtdeterministisch, sowie deren Äquivalenz zu OPGs bewiesen.

2.1 Operatorpräzedenzgrammatik

Eine kontextfreie Grammatik (kfG) ist ein 4-tupel $G = (N, \Sigma, P, S)$, wobei N die Menge der Nichtterminalsymbole, Σ die Menge der Terminalsymbole, P die Menge der Produktionsregeln und S das Startsymbol bezeichnet. Folgende Namenskonventionen werden im weiteren Verlauf verwendet: Kleine lateinische Buchstaben am Anfang des Alphabets $a, b, ...$ bezeichnen einzelne Terminalsymbole; Spätere, kleine lateinische Buchstaben $u, v, ...$ bezeichnen Terminalstrings; Große lateinische Buchstaben $A, B, ...$ stehen für Nichtterminalsymbole und griechische Buchstaben $\alpha, \beta, ...$ für beliebige Strings über $N \cup \Sigma$. Sofern nicht explizit eingeschränkt können Strings auch leer sein.
Weiterhin haben Produktionsregeln die Form $A \to \alpha$ mit der *leeren Regel* $A \to \epsilon$. *Umbenennende* Regeln haben nur ein Nichtterminalsymbol als rechte Seite . Eine *direkte Ableitung* wird mit $\Rightarrow$ beschrieben, eine beliebige Anzahl von Ableitungen mit $\overset{*}{\Rightarrow}$.
Eine Grammatik heisst *reduziert*, wenn jede Regel aus P benutzt werden kann um einen String aus Σ^* zu erzeugen. Sie ist *invertierbar*, wenn keine zwei Regeln identische rechten Seiten haben.
Eine Regel ist in *Operatorform*, wenn ihre rechte Seite keine benachbarten Nichtterminale hat. Entsprechend heisst eine Grammatik, die nur solche Regeln beinhaltet *Operatorgrammatik* (OG). Jede kfG $G = (N, \Sigma, P, S)$ kann in eine äquivalente Operatorgrammatik $G' = (N', \Sigma, P', S)$ umgewandelt werden [siam 25, 38].Die folgenden Definitionen gelten für Operatorpräzedenzgrammatiken (OPGs).

Definition 2.1 *Für eine OG G und ein Nichtterminal A sinddie linken und rechten Terminalmengen definiert als*

$$\mathcal{L}_G(A) = \{a \in \Sigma | A \overset{*}{\Rightarrow} Ba\alpha\},\ \mathcal{R}_G(A) = \{a \in \Sigma | A \overset{*}{\Rightarrow} \alpha aB\}$$

mit $B \in N \cup \{\epsilon\}$

Auf den Namen der Grammatik G kann verzichtet werden, wenn der Kontext klar ist. Eines der wichtigsten Merkmale von Operatorpräzedenzgrammatiken ist die Definition von drei binären Operatorpräzedenzrelationen.

Definition 2.2 (Präzedenzrelationen)
Gleiche Präzedenz: $a \doteq b \Leftrightarrow \exists A \to \alpha a B b \beta, B \in N \cup \{\epsilon\}$
Übernimmt Präzedenz: $a \gtrdot b \Leftrightarrow \exists A \to \alpha D b \beta, D \in N$ *and* $a \in \mathcal{R}_G(D)$
Gibt Präzedenz ab: $a \lessdot b \Leftrightarrow \exists A \to \alpha a D \beta, D \in N$ *and* $b \in \mathcal{L}_G(D)$

Es muss beachtet werden, dass diese Relationen im Gegensatz zu ähnlichen arithmetischen Relationen (<,>,=) keine transitiven, symmetrischen oder reflexiven Eigenschaften vorliegen. Weiterhin schließt die Gültigkeit einer Relation die andere nicht aus, sodass zB. sowohl $a \lessdot b$ als auch $a \doteq b$ gelten kann. Für eine OG G kann eine Operatorpräzedenzmatrix (OPM) $M = OPM(G)$ als $|\Sigma| \times |\Sigma|$ erstell werden, die für jedes geordnete Paar (a,b) die Menge M_{ab} der Operatorpräzedenzrelationen beinhaltet. Für solche Matrizen sind Inklusion und Vereinigung natürlich definiert.

Definition 2.3 *Eine OG G ist eine OPG oder auch FG gdw. M = OPM(G) eine konfliktfreie Matrix ist.*
Also: $\forall a, b, |M_{ab}| \leq 1$
Eine OPL ist eine Sprache, die durch eine OPG gebildet wird.

Für solche OPMs werden nun weitere Namenskonventionen vereinbart. Zwei Matrizen sind *kompatibel*, wenn ihre Vereinigung konfliktfrei ist. Eine Matrix heisst *total*, wenn gilt $\forall a, b : M_{\mathrm{ab}} \neq \emptyset$. Für eine OPG wird die *Fischer Normalform (FNF)* definiert.

Definition 2.4 (Fischernormalform) *Eine OPG G ist in Fischer Normalform gdw. gilt:*

- *G ist invertierbar*
- *G hat keine leeren Regeln außer dem Startsymbol, sofern dies nicht weiter verwendet wird*
- *G hat keine umbenennenden Regeln*

Zu jeder OPG kann eine äquivalente Grammatik in FNF gebildet werden. [ASDFGH] In Ausblick auf die Operatorpräzedenzautomaten erweitern wir die OPM um ein Symbol $\# \notin \Sigma$, welches Start und Ende eines Strings bezeichnet. Dieses Symbol beeinflusst die Grammatik nicht weiter, da für jedes Terminal gilt: $\forall a \in \Sigma : \# \lessdot a$ und $a \gtrdot \#$. Ferner gilt: $M_{\# \#} = \{\doteq\}$.

Definition 2.5 *Ein OP Alphabet ist ein Paar* (Σ, M) *mit:*
Σ *ist ein Alphabet*
M ist eine konfliktfreie OPM, erweitert zu einem $|\Sigma \cup \{\#\}|^2$ *Array, das zu jedem geordneten Paar (a,b) höchstens eine OP Relation enthält.*

Zum Schluss sollte noch eine Einschränkung bezüglich der $\doteq$-Relation getroffen werden. Aus 2.2 folgt, dass bei einer Regel $A \to A_1 a_1 ... A_{\mathrm{n}} a_{\mathrm{n}} A_{\mathrm{n}+1}$, bei der alle A_{i} potenziell fehlen können, die Relationen $a_1 \doteq a_1 \doteq ... \doteq a_{\mathrm{n}}$ gebildet werden. Problematisch wird dies, wenn die $\doteq$-Relation *zyklisch* ist dh. es gibt

$a_1, a_2, ..., a_\mathrm{m} \in \Sigma (m \geq 1)$, sodass $a_1 \doteq a_2 \doteq ... \doteq a_\mathrm{m} \doteq a_1$. Das führt dazu, dass die Länge der rechten Seite einer Regel keine Begrenzung hat. Wie in meinen Quellen gehe ich auch davon aus, dass die $\doteq$-Relation azyklisch ist. Dies kann einen Einfluss auf die Konstruktion mancher Grammatiken haben, allerdings betrifft es keine der hier verwendeten Grammatiken.
Nachdem hier die Grundlagen für die OPGs geschaffen wurden geht es weiter mit den Operatorpräzedenzautomaten.

2.2 Operatorpräzedenzautomaten

Die Operatorpräzedenzautomaten(OPA) verhalten sich ähnlich wie Pushdown-Automaten, aber erkennen genau die Operatorpräzedenzsprachen. OPAs arbeiten Bottom-Up, sind aber deutlich einfacher als zB. LR(k) Parser.

Definition 2.6 (Operatorpräzedenzautomat) *Ein nichtdeterministischer OPA ist ein 6-Tupel* $A = (\Sigma, M, Q, I, F, \delta)$ *mit*

- (Σ, M) *ist ein OP Alphabet*
- Q *ist eine Menge von Zuständen*
- $I \subseteq Q$ *ist die Menge der Startzustände*
- $F \subseteq Q$ *ist die Menge der akzeptierenden Zustände*
- $\delta : Q \times (\Sigma \cup Q) \to \mathcal{P}(Q)$ *ist die Übergangsfunktion, die aus drei Teilfunktionen besteht:*
  $\delta_{shift} : Q \times \Sigma \to \mathcal{P}(Q)$ $\delta_{push} : Q \times \Sigma \to \mathcal{P}(Q)$ $\delta_{pop} : Q \times Q \to \mathcal{P}(Q)$

Wie bei den meisten Automaten, bietet sich auch bei OPAs an eine graphische Darstellung einzuführen. Dabei werden die Zustände als Kreis mit einer Namenskennzeichnung dargestellt. Im weiteren Verlauf der Arbeit werden die Zustände meistens mit q_i, q_i oder r_i bezeichnet.
Die drei Transitionen werden mit verschiedenen Arten von Pfeilen dargestellt: Ein normaler Pfeil von q nach p mit einem Terminal a als Beschriftung steht für eine Push-Transition $p \in \delta_\mathrm{push}(q, a)$. Ein gestrichelter Pfeil steht für eine Shift-Transition $p \in \delta_\mathrm{shift}(q, a)$. Ein doppelter Pfeil mit einem Zustand r als Beschriftung steht dementsprechend für einen Pop-Move $p \in \delta_\mathrm{pop}(q, r)$
Als nächstes wird der Stack des Automaten definiert. Für die Elemente des Stacks definieren wir $\Gamma = \Sigma \times Q$ und erweitern dies um das Symbol für den leeren Stack $\Gamma' = \Gamma \cup \{\bot\}$. Elemente von Γ' haben die Form $[a, q]$ oder $\bot$.
Für Γ' werden ein paar Hilfsfunktionen einegführt: $symbol([a, q]) = a$, $symbol(\bot) = \#$ und $state([a, q]) = q$. Der Stack ist ein String der Form $\Pi = \bot\pi_1\pi_2...pi_\mathrm{n}$ mit $\pi_\mathrm{i} \in \Gamma$. Die symbol-Funktion wird auf Strings erweitert, indem das Symbol des letzten Elements ausgegeben wird: $symbol(\Pi) = symbol(\pi_\mathrm{n})$.
Eine *Konfiguration* eines OPA ist ein Tripel $C = (\Pi, q, w)$, wobei $\Pi \in \bot\Gamma^*$ den aktuelle Stack, $q \in Q$ den Zustand und $w \in \Sigma * \#$ das (übrige) Eingabewort bezeichnet.

Bei einem *Lauf* des Automaten handelt es sich um eine endliche Folge von Transitionen/Moves $C_1 \vdash C_2$. Die drei verschiedenen Moves sind wie folgt definiert:

Definition 2.7 (Übergangsfunktionen)
push move: if $symbol(\Pi) \lessdot a$ *then* $(\Pi, p, ax) \vdash (\Pi\,[a, p]\,, q, x)$ *mit* $q \in \delta_{push}(p, a)$
push move: if $a \doteq b$ *then* $(\Pi\,[a, p]\,, q, bx) \vdash (\Pi\,[b, p]\,, r, x)$ *mit* $r \in \delta_{shift}(q, b)$
pop move: if $a \gtrdot b$ *then* $(\Pi\,[a, p]\,, q, bx) \vdash (\Pi, r, bx)$ *mit* $r \in \delta_{pop}(q, p)$

Einfach gesagt bedeutet das: Bei einem Push-Move wird der Stack aufgebaut, während das Eingabewort weiter verbraucht wird. Der Shift-Move verarbeitet ebenfalls das Eingabewort, tauscht allerdings nur das Symbol im vorderen Stackelement aus. Der Pop-Move arbeitet dann den Stack ab. Shift- und Pop-Moves werden bei einem leeren Stack nicht ausgeführt.
Um den Automaten zu vervollständigen muss die akzeptierte Sprache definiert werden. Eine Konfiguration $(\bot, q_{\mathrm{I}}, w\#)$ mit q_{I} *inI* heisst *initial* und eine Konfiguration $(\bot, q_{\mathrm{F}}, \#)$ mit $q_{\mathrm{F}} \in F$ heisst akzeptierend.

Definition 2.8 (Akzeptanz der Sprache) *Die Sprache, die von einem OPA A erkannt wird ist definiert als*
$L(A) = \left\{ w | (\bot, q_I, w\#) \overset{*}{\vdash} (\bot, q_F, \#), q_I \in I, q_F \in F \right\}$

2.3 Äquivalenz von OPG und OPA

3 (Abschluss-) Eigenschaften von OPLs

3.1 Determistischer vs. Nondeterministischer OPA

3.2 Abgeschlossenheit

3.3 Visibly Pushdown Sprache als Teilklasse

4 Implementation of OPLs

5 Conclusion