Lektion 18

Kellerautomaten können für die effiziente, praktische Syntaxanalyse nicht direkt verwendet werden, da sie im allgemeinen nichtdeterministisch sind. Wir betrachten im Folgenden ein Syntaxanalyseverfahren mit sog. vorausschauenden Parsern, das das Grundprinzip der Top-Down-Analyse aufgreift, aber so modifiziert, dass sich ein effizientes, deterministisches Syntaxanalyse-Verfahren ergibt (was allerdings nicht für jede Grammatik verwendbar ist). Dazu müssen zunächst verschiedene Eigenschaften der kontextfreien Grammatik bestimmt werden.

Kap. 11 Effiziente Top-Down-Syntaxanalyse

Inhalt

- Effiziente Top-Down-Syntaxanalyse mit vorausschauenden Parsern
- Grammatik-Eigenschaften für die Top-Down-Analyse
- Berechnung der Voraussschautabelle
- LL(1)-Syntaxanalyse und LL(1)-Grammatiken
- Grammatikumformungen
- Ausblick: Parser-Generatoren

11.1 Effiziente Syntaxanalyse

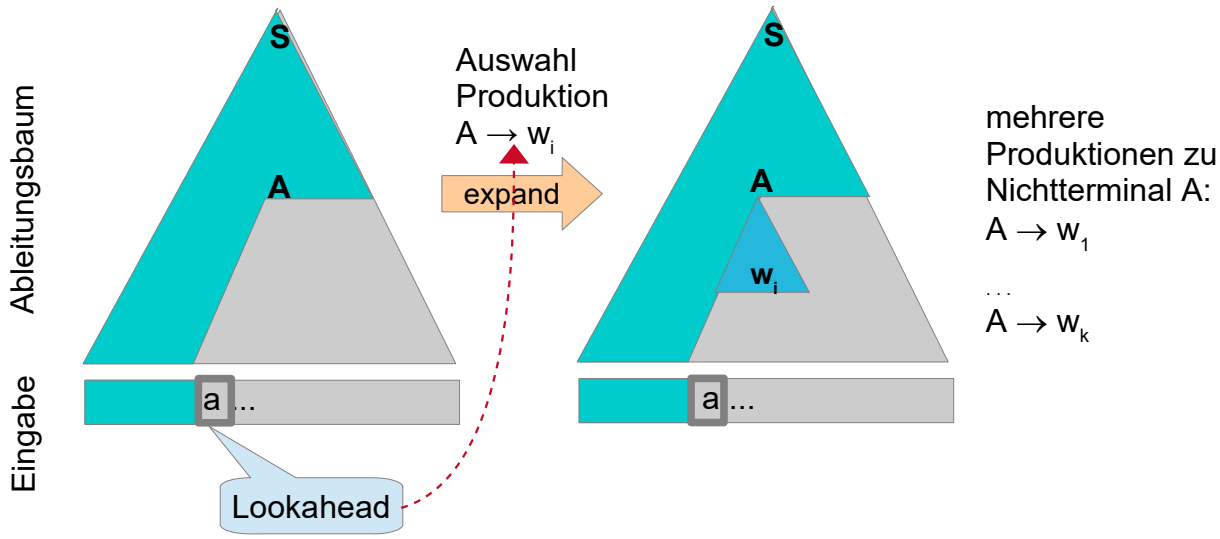
Das Prinzip der Top-Down-Syntaxanalyse wurde bereits im vorigen Kapitel vorgestellt. Es wurde gezeigt, wie zu jeder kontextfreien Grammatik ein Kellererautomat definiert werden kann, der eine Top-Down-Syntaxanalyse für die Grammatik realisiert. Allerdings sind diese Kellerautomaten nichtdeterministisch, da sie jeweils die passende Produktion für Expand-Schritte "raten" müssen, sofern es für ein nichtterminales Symbol mehrerer Produktionen gibt.

Top-Down-Analyse mit vorausschauenden Parsern

Soll die Top-Down-Syntaxanalyse als effizientes, deterministisches Verfahren verwendet werden können, muss deshalb bei den Expand-Schritten angesetzt werden. Um auszuwählen, welche Produktion zum Expandieren verwendet werden soll, wird die sog. *Vorausschau* (*Lookahead*) berücksichtigt, d.h. welches Zeichen in

der Eingabe als nächstes kommt. Abhängig vom aktuellen Lookahead-Zeichen wird dann bestimmt, welche der möglichen Produktionen für einen Expand-Schritt zu nehmen ist. Wir betrachten hier immer nur einen Lookahead von einem Zeichen, eine Erweiterung des Verfahrens auf mehrere Zeichen Vorausschau ist aber auch möglich.

Folgende Abbildung veranschaulicht das Prinzip der Top-Down-Analyse mit Vorausschau.



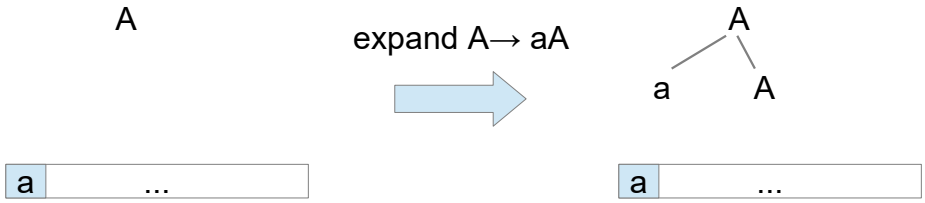
Beispiel 11.1 - Verwendung der Vorausschau

Es sei folgende Grammatik gegeben.

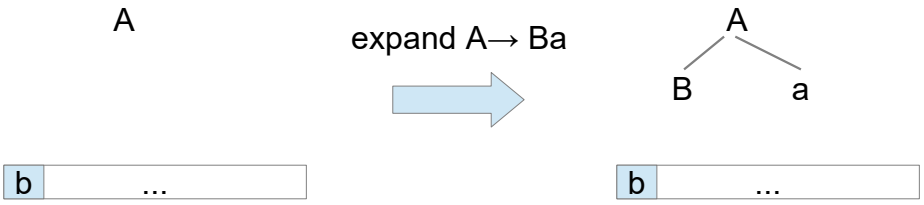
$$\begin{aligned} A &\rightarrow aA \\ &\mid Ba \\ B &\rightarrow bc \end{aligned}$$

Wir nehmen an, dass bei der Top-Down-Analyse ein A erwartet wird und expandiert werden muss.

- Beginnt die noch zu analysierende Eingabe mit dem Zeichen a (d.h. Lookahead ist a), dann ist offensichtlich, dass mit Produktion A → aA expandiert werden muss.



- Beginnt die noch zu analysierende Eingabe mit dem Zeichen b, dann kann nur die Produktion A → Bb richtig sein, denn aus B kann bc abgeleitet werden, was mit b beginnt.



- Beginnt die noch zu analysierende Eingabe mit dem Zeichen c, dann liegt garantiert ein Syntaxfehler vor, denn alles, was aus A abgeleitet wird, kann nur mit a oder b beginnen.

Vorausschautabelle für die Top-Down-Syntaxanalyse

Die Information, welche Produktion beim Expandieren genommen werden muss, kann in einer Tabelle dargestellt werden. Für jedes Nichtterminal ist für das nächste *Lookahead*-Zeichen eingetragen, welche Produktionen dafür in Frage kommen.

Bei der Vorausschautabelle muss auch berücksichtigt werden, dass das Ende der Eingabe erreicht sein kann. Es wird in der Tabelle deshalb eine Spalte angegeben, die mit $ als Kennzeichnung für das Eingabeende versehen ist.

Beispiel 11.2 - Vorausschautabelle

Die Vorausschautabelle für die Grammatik aus Beispiel 11.1 sieht so aus:

Lookahead *Nichtterminal*	*a*	*b*	*c*	*$ (Eingabe-ende)*
A	*A→ aA*	*A→ Ba*	–	–
B	–	*B → bc*	–	–

Anmerkung

- In einer Vorausschautabelle können in einer Zeile zu einem Nichtterminal *A* somit immer nur A-Produktionen enthalten sein.
- Ist ein Eintrag in der Tabelle leer, heißt das, dass in der entsprechenden Situation ein Syntaxfehler erkannt wurde, die Eingabe also nicht zur Sprache der Grammatik gehört. Enthält eine Zelle mehrere Produktionen, ist keine effiziente Analyse möglich, da nicht eindeutig bestimmt ist, welche der

Produktionen zu verwenden ist.

Steht eine solche Vorausschautabelle zur Verfügung, bei der an jeder Position höchstens eine Produktion eingetragen ist, ist eine effiziente Top-Down-Analyse möglich: Im Unterschied zum Kellerautomaten muss bei den Expand-Schritten nicht mehr "geraten" werden, welche Produktion zu nehmen ist, sondern es kann einfach in der Vorausschautabelle nachgeschlagen werden, welche Produktion zu verwenden ist.

11.2 Grammatik-Eigenschaften

Um die Vorausschautabelle für eine Grammatik bestimmen zu können, werden verschiedene Eigenschaften der nichtterminalen Symbole benötigt.

Das wichtigste Kriterium zu Auswahl passender Produktionen ist die sog. *First-Menge,* die angibt, mit welchen terminalen Symbolen die Zeichenfolgen beginnen können, die aus dem Nichtterminal abgeleitet werden können.

Etwas komplizierter wird es, wenn Nichtterminale zum leeren Wort ε abgeleitet werden können. Das wird als die sog. *nullable*-Eigenschaft bezeichnet. Dann ist manchmal nicht nur die First-Menge wichtig, sondern auch, was nach dem ggf. leeren Teil kommt, der aus dem Nichtterminal abgeleitet werden kann, die sog. *Follow*-Menge.

Insgesamt benötigen wir folgende drei Eigenschaften, für die nachfolgend genauer beschrieben wird, wie sie definiert sind und wie sie berechnet werden können:

- **nullable(A)**: Kann das nichtterminale Symbol *A* zum leeren Wort ε abgeleitet werden?
- **First(A)**: Mit welchen terminalen Symbolen kann ein Wort beginnen, das aus dem Nichtterminal A abgeleitet wird?
- **Follow(A)**: Welche terminale Symbole können nach dem Teilwort folgen, das aus A abgeleitet wird?

11.2.1 nullable-Eigenschaft

Von den Grammatikeigenschaften am einfachsten zu verstehen und zu berechnen ist die *nullable*-Eigenschaft.

Definition 11.3 - nullable-Eigenschaft

Bei einer kontextfreien Grammatik $G = (N, T, P, S)$ ist die Eigenschaft **nullable(*A*)** ist für ein Nichtterminal $A \in N$ genau dann wahr, wenn A zum

leeren Wort ε abgeleitet werden kann, d.h. falls gilt $A \Rightarrow^* \varepsilon$.

Welche nichtterminalen Symbole die nullable-Eigenschaft haben, kann mit dem nachfolgenden iterativen Algorithmus berechnet werden, der folgende Eigenschaften verwendet.

- Gibt es für ein nichtterminales Symbol A eine ε-Produktion, dann ist A nullable.
- Ist beispielsweise schon bekannt, dass A und B die nullable-Eigenschaft haben und gibt es eine Produktion C → AB, dann gilt auch nullable(C).
- Für terminale Symbole ist die nullable-Eigenschaft niemals wahr.

Algorithmus 11.4 - Berechnung der nullable-Eigenschaft

1. Initialisierung:
Für alle Nichtterminale *A* mit **ε-Produktion** $A \rightarrow \varepsilon$
setze nullable(*A*) = true

2. Wiederhole nacheinander für jede Nicht-ε-Produktion der Grammatik:
Falls die Produktion die Form
$A \rightarrow Y_1 Y_2 ... Y_k$ hat, wobei die Y_i nichtterminale Symbole sind
und falls nullable(Y_i) für alle i von 1 bis *k* **dann**
setze nullable(*A*) = true

so lange, bis sich die nullable-Eigenschaft für kein Symbol mehr ändert.

Die nullable-Eigenschaft kann in analoger Weise auf Folgen von Symbolen erweitert werden.

Definition 11.5 - nullable-Eigenschaft für Symbolfolgen

Für ein Symbolfolge $w \in (N \cup T)^*$ *ist* **nullable(w)** *wahr*, falls *w* zum leeren Wort ε abgeleitet werden kann, d.h. wenn

$w \Rightarrow^* \varepsilon$

gilt.

Anmerkung

- Damit nullable(*w*) für eine Symbolfolge *w* gilt, muss jedes Symbol der Folge ein nichtterminales Symbol sein, das die nullable-Eigenschaft hat.
- Wenn eine Symbolfolge *w* mindestens ein terminales Symbol enthält, ist nullable(*w*) folglich nicht wahr.

Beispiel 11.6 - Berechnung von nullable()

Gegeben ist folgende Grammatik:

$S \rightarrow d \mid ABS$
$A \rightarrow BC \mid a$
$B \rightarrow \varepsilon \mid bCe$
$C \rightarrow \varepsilon \mid c$

Berechnung der nullable-Eigenschaft:

(1) Initialisierung:

Es gibt folgende ε-Produktionen:

$B \rightarrow \varepsilon$ $\Rightarrow$ nullable(B) eintragen

$C \rightarrow \varepsilon$ $\Rightarrow$ nullable(C) eintragen

(2) Produktionen nur nichtterminalen Symbolen auf der rechten Seite untersuchen:

1. Durchgang:

$S \rightarrow ABS$ nullable(A) und nullable(B) schon bekannt, nullable(S) noch nicht bekannt
$\Rightarrow$ nichts für S eintragen

$A \rightarrow BC$ nullable(A) und nullable(B) schon bekannt,
$\Rightarrow$ nullable(A) eintragen

2. Durchgang:

$S \rightarrow ABS$ nullable(A) und nullable(B) schon bekannt, nullable(S) noch nicht bekannt
$\Rightarrow$ nichts für S eintragen

($A \rightarrow BC$ $\Rightarrow$ nullable(A) ist schon eingetragen)

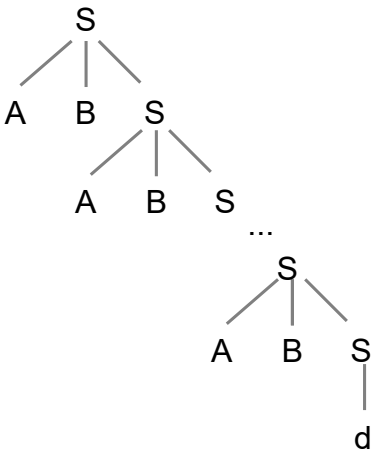
Keine Änderung im 2. Durchgang, Berechnung endet somit

	Initialisierung (ε-Produktionen)	1. Durchgang	2. Durchgang
S	-	-	-
A	-	ja	ja
B	ja	ja	ja
C	ja	ja	ja

Anmerkung

Ist das Ergebnis des vorigen Beispiels, dass S nicht nullable ist, plausibel?

Betrachtet man mögliche Ableitungsbäume, so sieht man, dass S so wie hier angegeben verwendet werden kann.

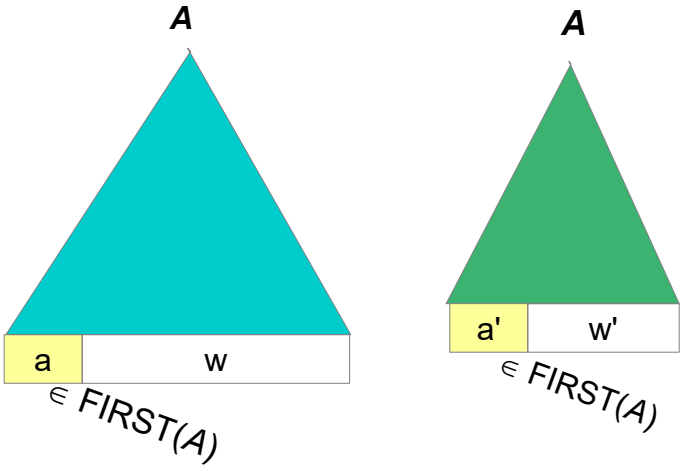


Man erkennt, dass alles, was aus S abgeleitet werden kann, immer ein d am Ende hat. Somit kann S nicht zu ε abgeleitet werden, ist also nicht nullable.

11.2.2 First-Mengen

Ist eine kontextfreie Grammatik $G = (N, T, P, S)$ gegeben, dann ist **First(*A*)** für ein Nichtterminal $A \in N$ die Menge aller terminalen Symbole $a \in T$, die *am Anfang* von Wörtern stehen können, die aus A ableitbar sind.

Veranschaulichung der First-Mengen



alle möglichen Ableitungsbäume mit A als Wurzel sind zu berücksichtigen!

Definition 11.7 - First-Mengen

Für die nichtterminalen Symbole $A \in N$ einer kontextfreie Grammatik $G = (N, T, P, S)$ sind die First-Mengen folgendermaßen definiert.

$$\textbf{First}(A) = \{ a \in T \mid \text{es existiert ein } w \in (N \cup T)^*, \text{ so dass } A \Rightarrow^* aw \}$$

Beispiel 11.8 - First-Mengen

Für die Grammatik

$A \rightarrow$ aA
| Ba
$B \rightarrow$ bc

gilt:

First(B) = { b }
First(A) = { a, b }

- Offensichtlich ist First(B) = { b }, da aus B nur die Zeichenfolge bc abgeleitet werden kann, die mit einem b beginnt.
- Aus A kann aA abgeleitet werden, was mit a beginnt, somit ist a in First(A) enthalten. Das Zeichen b ist auch in First(A) enthalten, da A nach Ba abgeleitet werden kann und B mit b beginnen kann (b ist in First(B) enthalten).

Der Begriff der First-Menge kann auch auf ganze Folgen von terminalen und nichtterminalen Symbolen erweitert werden.

Definition 11.9 - First(w) für Wörter

Für eine kontextfreie Grammatik G = (N, T, P, S) und Symbolfolgen $w \in (N \cup T)^*$ sind die First-Mengen First(w) folgendermaßen definiert:

First(*w*) = { $a \in T$ | es existiert ein $w' \in (N \cup T)^*$, so dass $w \Rightarrow^* aw'$ }

Aufbauend auf der First-Menge für einzelne nichtterminale Symbole kann die First-Menge für Folgen von terminalen und nichtterminalen Zeichen folgendermaßen rekursiv berechnet werden:

Berechnung 11.10 - First(w) für Wörter

Ist eine kontextfreie Grammatik G = (N, T, P, S) und ein Symbolfolge $w \in (N \cup T)^*$ gegeben, dann wird **First(*w*)** so berechnet:

(1) Falls $w = \varepsilon$:
First(ε) = { }

(2) Falls $w = aw'$, wobei $a \in T$, $w' \in (N \cup T)^*$
First(aw') = { a }

(3) Falls w = Aw', wobei $A \in N$ und *nicht* nullable(A), $w' \in (N \cup T)^*$,
First(Aw') = First(A)

(4) Falls $w = Aw'$, wobei $A \in N$ und nullable(A), $w' \in (N \cup T)^*$,
First(Aw') = First(A) $\cup$ First(w')

- zu (2): Beginnt eine Folge mit einem terminalen Symbol, z.B. aBC, dann ist nur dieses Symbol in der First-Menge enthalten.
- Beginnt ein Wort mit einem nichtterminalen Symbol, z.B. ABC, hängt es davon ab, ob A nullable ist oder nicht.
  - (3) Falls A nicht nullable ist, ist nur die First-Menge First(A) zu berücksichtigen.
  - (4) Falls A nullable ist, dann ist sowohl die First-Menge des Symbols A als auch die First-Menge des danach folgenden Restworts BC zu berücksichtigen.

Beispiel 11.11 - First(w) für Symbolfolgen

Folgende Grammatik ist gegeben:

$B \rightarrow \quad \varepsilon \mid bCe$

$C \rightarrow \quad \varepsilon \mid c$

Es soll

(1) First(aBC)

(2) First(BaC)

(3) First(BCd)

berechnet werden

Folgende Grammatikeigenschaften sind direkt zu erkennen:

	nullable	**First**
B	ja	b
C	ja	c

(1) Berechnung von First(aBC):

First(aBC) = { a }

d.h. ein davon abgeleitetes Wort kann nur mit a anfangen.

(2) Berechnung von First(BaC):

First(BaC) = First(B) ∪ First(aC) da nullable(B)

= {b} ∪ {a} da aC mit Terminal beginnt.

= {a, b}

B kann mit b beginnen, aber da B auch ε ergeben kann, kann ein abgeleitete Wort außerdem mit dem nachfolgenden a beginnen.

(3) Berechnung von First(BCd):

First(BCd) = First(B) ∪ First(Cd) da nullable(B)

= {b} ∪ First(C) ∪ First(d) da nullable(C)

= {b} ∪ { c } ∪ {d}

= {b, c, d}

d.h. aus BCd können Wörter abgeleitet werden, die mit b, c oder d beginnen.

Der folgende iterative Algorithmus zur Berechnung der First-Mengen für die nichtterminalen Symbole einer Grammatik beruht darauf, dass man immer wieder die Produktionen durchgeht und für jede Produktion

$A \rightarrow w$

jeweils bestimmt, was (nach bisher berechnetem Wissen) in First(w) enthalten ist, d.h. womit Wörter beginnen können, die aus w abgeleitet werden. Diese Zeichen werden dann auch in First(A) mit aufgenommen.

Algorithmus 11.12 - Berechnung der First-Mengen für Nichtterminale

1. Initialisierung: für alle Nichtterminale A initialisiere First(A) = { }.

2. Wiederhole

Gehe alle Produktionen der Grammatik nacheinander durch:

Für die jeweilige Produktion $A \rightarrow w$

nimm First(w) zu First(A) dazu.

solange bis sich keine First-Menge mehr ändert.

First(…) bezeichnet hierbei jeweils die bisher berechneten Einträge für die First-Menge. Am Ende der Ausführung entspricht das dann der kompletten First-Menge.

Beispiel 11.13 - First-Mengen für nichtterminale Symbole berechnen

Gegeben Sei folgende Grammatik, siehe Aufgabe 11.6

S → d | ABS
A → BC | a
B → ε | bCe
C → ε | c

Es sollen die First-Mengen für alle Nichtterminale berechnet werden.

- Ausgangssituation nach Initialisierung:

	nullable	*First*
S	-	
A	ja	
B	ja	
C	ja	

- 1. Durchgang:

S → d: First(d) = {d} nimm d zu First(S) dazu (1)
S → ABS: First(ABS) = {d}, nullable(A) und nullable(B) gilt, First(A) und First(B) noch leer, d in First(S) enthalten

A → BC:	First(BC) = {}	First(B) und First(C) bisher leer
A → a:	First(a) = {a}	nimm a zu First(A) dazu (2)
B → bCe	First(bCe) = {b}	nimm b zu First(B) dazu (3)
C → c	First(c) = {c}	nimm c zu First(C) dazu (4)

	nullable	*First*
S	-	d(1)
A	ja	a(2)
B	ja	b(3)
C	ja	c4)

- 2. Durchgang: (bei Produktionen, die mit einem Terminal beginnen, kann nichts neues dazukommen, sie werden daher nicht nochmal betrachtet).

S → ABS:	First(ABS) = {a, b, d},	nimm a, b, d zu First(S) dazu (5)
A → BC:	First(BC) = {b, c}	nimm b, c zu First(A) dazu (6)

	nullable	*First*
S	-	d(1), a, b(5)
A	ja	a(2), b, c (6)
B	ja	b(3)
C	ja	c4)

- 3. Durchgang:

S → *AB*S:	First(ABS) = {a, b, c, d},	c kommt bei First(S) dazu (7)
A → *BC:*	First(BC) = {b, c}	keine Änderung

	nullable	*First*
S	-	d(1), a, b(5), c(7)
A	ja	a(2), b, c (6)
B	ja	b(3)
C	ja	c4)

- 4. Durchgang:

S → *AB*S:	First(ABS) = {a, b, c, d},	keine Änderung
A → *BC:*	First(BC) = {b, c}	keine Änderung

Die Berechnung ist nun fertig, da kein neuer Eintrag dazukam.

Aufgabe 11.14 - First-Mengen für Wörter

Geben Sie

(1) First(BaS)

(2) First(ABS)

(3) First(BC)

für die Grammatik aus dem vorangehenden Beispiel 11.13 an.

Aufgabe 11.15 - First-Mengen berechnen

Berechnen Sie die First-Mengen für folgende Grammatik:

$$
\begin{aligned}
S &\rightarrow AS \mid d \\
A &\rightarrow Bc \mid aA \\
B &\rightarrow \varepsilon \mid b
\end{aligned}
$$

Zusammenfassung zu Lektion 18

Diese Fragen sollten Sie nun beantworten können

- Was versteht man unter einem vorausschauenden Parser?
- Was bedeutet die nullable-Eigenschaft für ein nichtterminales Symbol *A*?
- Wie kann die nullable-Eigenschaft für die nichtterminalen Symbole einer kontextfreien Grammatik berechnet werden?
- Was bedeutet die First-Menge für einzelne nichtterminale Symbole oder ganze Symbolfolgen?
- Wie können die First-Mengen der nichtterminalen Symbole berechnet werden?