Prof. Dr. Arnd Poetzsch-Heffter
Dr.-Ing. Jan Schäfer

Technische Universität Kaiserslautern
Fachbereich Informatik
AG Softwaretechnik

Übungsblatt 5: Übersetzer und sprachverarbeitende Werkzeuge (SS 2011)

Hand Out: 18. Mai 2011
Hand In: 25. Mai 2011

Aufgabe 1 Berechnung von FIRST_1 und FOLLOW_1

Der folgende Algorithmus berechnet die FIRST_1 und FOLLOW_1-Mengen für eine gegebene kontextfreie Grammatik (siehe Appel, s. 50). Es wird dabei das Hilfsprädikat nullable(X) verwendet, das besagt, ob von X das leere Wort abgeleitet werden kann. Eine ϵ-Ableitung entspricht hier einer Ableitung $X \rightarrow .\ \alpha$ steht hier für entweder ein Terminal oder Nicht-Terminal Symbol. Initial gilt für alle X, nullable$(X) =$ false, $\text{FIRST}[X] = \text{FOLLOW}[X] = \emptyset$.

for each terminal symbol a
  $\text{FIRST}[a] \leftarrow \{a\}$
repeat
  for each production $X \rightarrow \alpha_1 \alpha_2 \cdots \alpha_k$
    if $\alpha_1 \cdots \alpha_k$ are all nullable (or if $k = 0$)
      then nullable$[X] \leftarrow$ true
    for each i from 1 to k, each j from $i+1$ to k
      if $\alpha_1 \cdots \alpha_{i-1}$ are all nullable (or if $i = 1$)
        then $\text{FIRST}[X] \leftarrow \text{FIRST}[X] \cup \text{FIRST}[\alpha_i]$
      if $\alpha_{i+1} \cdots \alpha_k$ are all nullable (or if $i = k$)
        then $\text{FOLLOW}[\alpha_i] \leftarrow \text{FOLLOW}[\alpha_i] \cup \text{FOLLOW}[X]$
      if $\alpha_{i+1} \cdots \alpha_{j-1}$ are all nullable (or if $i + 1 = j$)
        then $\text{FOLLOW}[\alpha_i] \leftarrow \text{FOLLOW}[\alpha_i] \cup \text{FIRST}[\alpha_j]$
until FIRST, FOLLOW, and nullable did not change in this iteration

a) Der oben genannte Algorithmus berechnet nullable, FIRST und FOLLOW in verschränkter Art und Weise. Wie man leicht sieht, kann man die Berechnungen auch nacheinander durchführen. Verändern Sie den Algorithmus so, dass zuerst nullable, dann FIRST und dann FOLLOW berechnet wird, wobei jeweils die vorherigen Ergebnisse in den darauf folgenden Berechnungen verwendet werden.

b) Wenden Sie Ihren umgeformten Algorithmus auf die folgende Grammatik Γ_1 an und berechnen Sie die FIRST_1 und FOLLOW_1 Mengen zu allen Nicht-Terminalsymbolen. Geben Sie auch den Lösungsweg an, indem Sie (sinnvolle) Zwischenergebnisse und Berechnungsschritte des Algorithmus angeben.

$$\Gamma_1 : \begin{array}{l} S \rightarrow A\# \\ A \rightarrow BAa \\ A \rightarrow \epsilon \\ B \rightarrow bBc \\ B \rightarrow AA \end{array}$$

c) Wie unterscheiden sich die FIRST_1- und FOLLOW_1-Mengen, die der Algorithmus berechnet, von denen der Vorlesung? Geben Sie die FIRST_1-Mengen aller Nicht-Terminale von Γ_1 an, nach der Definition in der Vorlesung.

d) Ist Γ_1 LL(1)? Warum?

e) Vereinfachen Sie den oben genannten Algorithmus für Grammatiken, bei denen es keine ϵ-Produktionen gibt (nullable$(X) =$ false, für alle Nicht-Terminale)

Aufgabe 2 Abstrakte Syntaxbäume aufbauen

Auf der Webseite zur Vorlesung finden Sie eine Datei `MiniJava.katja`, die einen AST für MiniJava angibt. Dieser ist auch unten angegeben. Bauen Sie gemäß dieser Spezifikation einen AST während des Parsens mit dem CUP-Parser auf.

An einigen Stellen weicht die Struktur des AST von der Struktur der Grammatikregeln ab. Überlegen Sie sich wie Sie damit umgehen. Sie können auch die Katja-Spezifikationen um weitere Sorten erweitern um Teilergebnisse während des Parsens besser behandeln zu können. Am Ende sollte jedoch eine Termpositionsstruktur entstehen, die der originalen Katja-Spezifikation entspricht. Wenn Sie andere Hilfssorten verwenden müssen Sie diese am Schluss heraus transformieren.