Compiler und Programmtransformation

Übung 4

(Parsing: Bottom-Up)

Henning Bordihn

Institut für Informatik und Computational Science

Universität Potsdam

1. Shift-Reduce-Parsing

Betrachten Sie die folgende kontextfreie Grammatik (mit Terminalzeichen *a* und *b,* Nichtterminalen *A*, *B* und Startsymbol *A*):

$A \rightarrow AB \mid ab$
$B \rightarrow aba$

1) Ermitteln Sie einen Shift-Reduce-Parse für die Eingabe *ababa* (*so wie auf den Folien 19 bis 28 aus „Parsing: Bottom-Up“*), wobei Ihr Parser folgende Strategien zur Konfliktlösung befolgt:

   • Shift-Reduce-Konflikt: Reduce hat Vorrang vor Shift, d.h., immer, wenn ein Reduce-Schritt möglich ist, führen Sie diesen aus, sonst führen Sie einen Shift-Schritt aus.

   • Reduce-Reduce-Konflikt: Reduce-Schritte werden in der Reihenfolge angewendet, wie sie in der Definition der Grammatik angegeben ist.

   • Ist das Ende der Eingabe erreicht und keine Reduktion auf das Startsymbol möglich, so erfolgt ein Backtrack-Schritt bis zum zuletzt ausgeführten Reduce-Schritt, der noch eine andere Aktion erlaubt.

2) Eine kontextfreie Grammatik heißt zyklenfrei, falls sie kein Nichtterminal A enthält, für das $A \Rightarrow^{+} A$ gilt. Erklären Sie, weshalb Grammatiken

   a) zyklenfrei und

   b) frei von ε-Produktionen

   sein müssen, damit der obige Shift-Reduce-Parser in jedem Fall erfolgreich angewendet werden kann.

2. LR-Parsing

1) Betrachten Sie erneut die Grammatik aus Aufgabe 1.1.

   a) Konstruieren Sie den zugehörigen LR(1)-Parser nach der Methode aus der Vorlesung.

   b) Wenden Sie den in Teil a) konstruierten Parser auf die Eingabe *ababa* an.

2) Betrachten Sie die folgende eindeutige kontextfreie Grammatik:

   $A \rightarrow aAa \mid bAb \mid \varepsilon$

   Handelt es sich um eine LR(1)-Grammatik? Wenn nicht, kann der lookahead geeignet vergrößert werden, so dass eine LR(k)-Grammatik für ein geeignetes $k > 1$ vorliegt?